SHIELD

SECURING AGAINST INTRUDERS AND OTHER THREATS THROUGH A NFV-ENABLED ENVIRONMENT

[H2020 - Grant Agreement No. 700199]

Deliverable D3.2

Updated specifications, design and architecture for the vNSF ecosystem

Editor	O. E. Segou (ORION)
Contributors	G. Gardikis (SPH), C. Xilouris, D. Christinakis (ORION), C. Fernandez, B. Gaston (i2Cat), E. Trouva, I. Aggelopoulos, A. Kourtis (NCSRD), L. Jacquin, H. Attak (HPELB), M. De Benedictis, A. Lioy (POLITO), F. Ferreira, R. Preto (Ubiwhere), A. A. Pastor, J. N. Mendoza (TID).
Version	1.0
Date	March 31th, 2018
Distribution	PUBLIC (PU)

Executive Summary

Following the work done in D2.1/D2.2, where the requirements of the SHIELD platform were elicited and the high-level design and architecture of the platform was exposed, a detailed study of the different components has been done in order to obtain the low-level architecture and design (subcomponent granularity), the specifications (transformation of the user requirements into technical requirements/specifications) and the implementation guide (technologies to use). This work has been divided into the two technical development work packages of SHIELD namely WP3 and WP4. Deliverable D3.1 "Specifications, design and architecture of the vNSF ecosystem" presented the first draft of the specifications and design of the components developed in WP3, namely:

i) the vNSF Store, which holds a registry of NS and vNSF-related information;
ii) the vNSF Orchestrator, which deploys and manages the lifecycle of the NSs and vNSFs;
iii) the monitoring vNSFs, which produce the information to detect the threats;
iv) the remediation vNSFs, which actuate and mitigate detected threats, and
v) the Trust Monitor, which verifies that both NSs and vNSFs, as well as other nodes from the infrastructure, are trusted at all times.

SHIELD's engineering process is based on two iterations of the requirements elicitation. The final specifications and design of these components is herein provided, based on the preliminary work in D3.1 and the updated requirements in D2.2. This work concludes the transformation of requirements into a high-level design and architecture, that later evolve to technical specifications. It also includes a list of legal and ethical compliance specifications, which serve to provide the user with the available information on each component. The requirements identified in D2.2 were categorised into:

- **platform functional (PF) requirements,** that detail the functionalities required by the platform,
- **platform non-functional (NF) requirements,** that detail the performance, ease of use and security of the platform
- **service functional (SF) requirements**, that describe the functionalities of the cybersecurity Network Services that the platform deploys
- **ethical and regulatory compliance (ERC) requirements** that focus on maintaining the platform's alignment with the EU regulatory landscape.

Based on these requirements, SHIELD herein defines the vNSF architecture blueprints, in which the common elements of a vNSF are defined and the available interfaces are depicted. It also defines how a Network Service (NS) is built from one or more vNSFs and how control and configuration is ensured. Both the architecture and the interfaces presented for vNSF comply with the ETSI NFV group recommendations and specifications [1].

SHIELD bases its developments on open source tools and evolves research results from previous projects, such as SONATA [2], the ETSI-supported OSM project [3], and the SECURED [4] project, for instance reusing the Third-party Verifier based on Open Attestation v1.7 [5], the Whitelist Database based on Apache Cassandra 2 [6] and the SDN-enabled switch attestation prototype [7]. The current document aims to bring these specifications together and highlight the organic links between components.

Table of Contents

1. INTRODUCTION

1.1. SHIELD project overview

This document presents the detailed architecture, design and specifications of the components involved in the Virtual Network Security Function (vNSF) ecosystem, within Work Package 3 (WP3). It summarises the work done in the first iteration of T3.1. This deliverable starts from the high-level architecture, design and requirements presented in D2.1; and provides specific details of the components' design, definition and their adequateness regarding the SHIELD requirements. SHIELD, as a Use-Case (UC) driven project, aims to cover the functionality required by the following three Use Cases (defined in D2.1 and briefly repeated here for the sake of completeness):

- *Use Case 1:* An Internet Service Provider (ISP) using SHIELD to secure its own infrastructure. This UC involves the ISPs deploying vNSFs in their network to detect security incidents and provide protection against them (Figure 1).
- *Use Case 2:* An ISP is leveraging SHIELD to provide advanced Security as a Service (SecaaS) services to its customers. This UC assumes that network security services (consisting of vNSFs), along with real-time incident detection and mitigation services, are offered as-a-Service to ISP clients, such as enterprises, public bodies, etc. (Figure 2).
- *Use Case 3:* Contributing to national, European and global security. This UC assumes that incident information is exposed, in a secure and private manner, to public cybersecurity authorities (Figure 3).

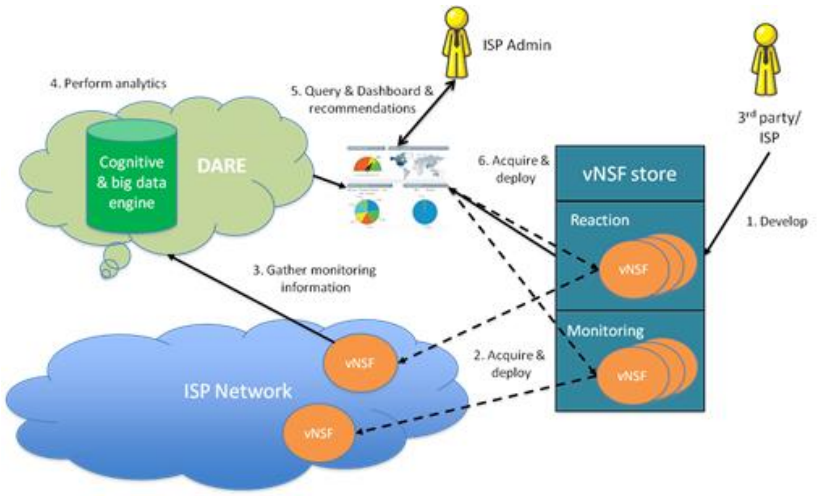


Figure 1: High-level picture of use case 1 focusing on the ISP network.

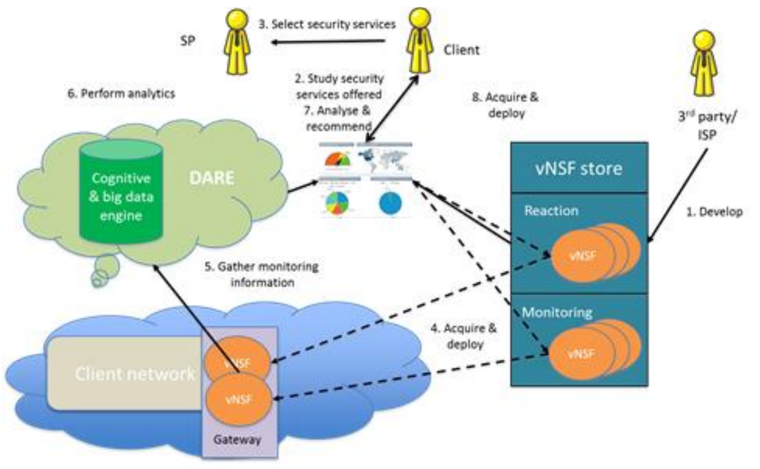


Figure 2: High-level picture of use case 2, focusing on SecaaS.

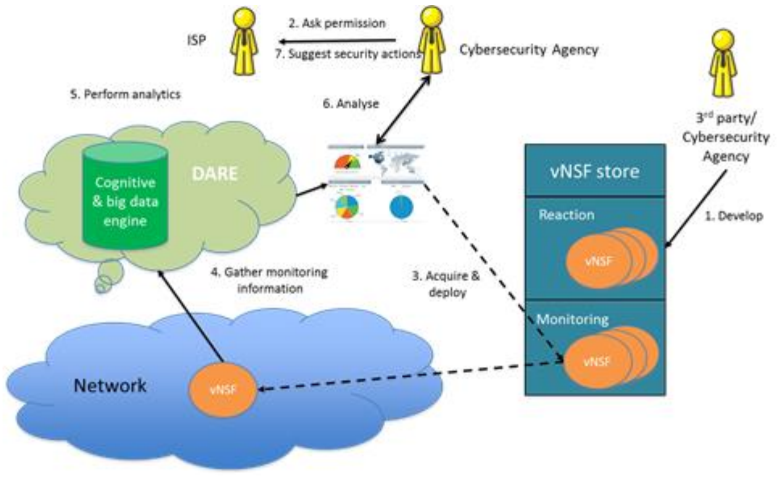


Figure 3: High-level picture of use-case 3, focusing on national, European and global security.

The high-level architecture defined in WP2 and reproduced on Figure 4 states that SHIELD consists of 6 main components; of these, WP3 deals with the vNSF Ecosystem, the vNSF Orchestrator (vNSFO), the Store and the Trust Monitor (TM). The other two components, the DARE and the Security Dashboard are developed in WP4 and D4.1/D4.2 provide their specifications.

Although the three use cases form the basis of the analysis, the resulting architecture, design, specifications and implementation have been elaborated to produce a unified and universal solution; i.e., a single cybersecurity solution that can be used for multiple purposes. The SHIELD platform provides the actors in the different use cases with different views and roles on the network. For example, while an ISP (use case 1) can view the big picture of the cybersecurity analysis and can directly deploy NSs in any location of the network; the ISP client (use case 2) only has access to a limited vision of the cybersecurity picture (information that is offered by the ISP and/or paid by the client) and can request the ISP for deployment of cybersecurity services (mapped to one or more NSs) in specific places of the network (i.e. to its gateways) in order to protect their own services. Cybersecurity agencies (use case 3) have a country or European-wide security view of the communication infrastructure, as well as the security threats and incidents that take place over this infrastructure, without having access to sensitive information that belongs to ISPs and their clients – which could reveal potential business plans or data.

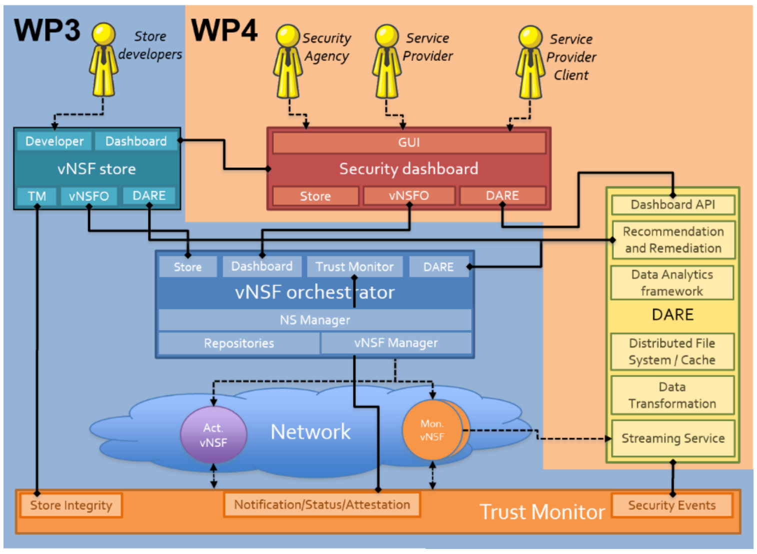


Figure 4: High-level architecture of SHIELD, with components per WP.

Based on these use cases and the requirements highlighted in Deliverables D2.1 [8] and D2.2 [9], the designed high-level architecture for the SHIELD platform is articulated around different components, illustrated in Figure 4 and described in more detail in this deliverable. From the point of view of the vNSF environment; the vNSF Store holds a record of Network Services (NS) and vNSF-related information and provides an endpoint to the developer for the onboarding of such services, which are later deployed by the vNSF Orchestrator into any given infrastructure and managed during its lifetime. Once deployed, vNSFs and NSs are verified by the Trust Monitor on bootstrap and at runtime, along with other nodes from the infrastructure; assessing their trustworthiness at all times. These core components, as part of WP3, are complemented by those in WP4: i) the DARE, storing and analysing the security logs and events provided by the running NSs and vNSFs; and ii) the Security Dashboard, presenting the results from the DARE to the operator. Both DARE and Security Dashboard components are detailed in deliverable D4.1 [10] and D4.2 [11] although, for the sake of providing a comprehensive deliverable, a summary of their function will be presented here.

Monitoring vNSFs inspect captured data and provide valuable information to the DARE. The network status is reported periodically, and all this data is centralised in the DARE. The data analytics framework (DARE subcomponent) analyses all the heterogeneous network information previously collected via monitoring vNSFs and the Trust monitor. It features cognitive and analytical components capable of predicting specific vulnerabilities and attacks. Finally, the remediation engine subcomponent of the DARE provides recommendations in the form of new network services or medium level policies (configurations of existing VNSFs) to remediate the detected threats. These recommendations and the attack information are displayed through the graphical user interface provided by the Security Dashboard component, which allows authenticated and authorized users to access SHIELD's functionalities. Privileged operators have also access to the deployment of specific NSs and to the monitoring information that provides an overview of the security status. Specific authorized users will therefore be able to visualise DARE recommendations and react through the Security Dashboard, the vNSFO by deploying new services (NS, VNSFs) if required, or configuring the existing services (NS, VNSFs) to mitigate the attack.

1.2. Scope of this document

SHIELD dedicates WP3 ("vNSFs ecosystem") to the technical work required towards achieving the following key goals: (a) To develop the SHIELD Network Services (NSs) and virtual Network Security Functions (vNSFs), along with the necessary infrastructure; (b) To develop the vNSF orchestration and management features; (c) To evaluate the cybersecurity capabilities of the SHIELD vNSFs and their compliance with EU regulations; (d) To develop the vNSF Store and plan its evolution from a repository to a viable marketplace concept; and (e) To explore the trusted computing infrastructure that attests the vNSFs and NFVI.

This document (D3.2 "Updated specifications, design and architecture of the vNSF ecosystem") details the final design choices regarding the vNSF and trusted computing infrastructure. During M1-M19, SHIELD has developed its NFVI, some key NSs and vNSFs and the attestation framework, which were first demonstrated during the Y1 review. D3.2 draws inputs from the following deliverables:

- D2.1 "Requirements, KPIs, design and architecture" [8] defines high-level requirements for the SHIELD platform and the overall architecture, including the KPIs to use in evaluation phase. D2.2 "Updated requirements, KPIs, design and architecture" [9] is the final, updated version of D2.1, which was drafted concurrently with this document.
- D3.1 "Specifications, design and architecture for the vNSF ecosystem" [12] contains the first version of the design and specifications for the SHIELD vNSFs, Orchestrator, Store and Trust monitor. This document builds upon D3.1 and provides the finalized specifications and design.
- D4.1 "Specifications, design and architecture for the usable information-driven engine" [10] contains the detailed design and specifications for SHIELD's DARE components, including analysis and remediation.
- D5.1 "Integration results of SHIELD HW/SW modules" [13] provides guidelines for the integration and testing of vNSF ecosystem components.

1.3. Organisation of this document

This document is organised as follows:

- **Chapter 1** (present chapter) serves as a basic introduction to this document and its scope;
- **Chapter 2** provides an overview of the design and architecture of the vNSF ecosystem.
- **Chapter 3** lists the specifications and implementation details for the vNSF ecosystem.
- **Chapter 4** discusses the regulatory and ethical compliance specifications for the vNSF ecosystem and is a new addition to this document;
- **Chapter 5** includes the validation guidelines for the vNSF ecosystem and discusses certification of vNSF appliances to strengthen future exploitation;
- **Chapter 6** concludes the document and lists future WP3 work;
- **Annexes A through F** provide further technical details regarding the implementation of the vNSF ecosystem, and
- **Annex G** lists the technical updates and the history of changes from D3.1 to D3.2.

2. Design and Architecture

2.1. Guiding Principles

SHIELD acknowledges that the future exploitation and evolution of its components is reliant on their adoption in real operational environments. This, however, hinges on the platform's **compliance with well-known standards and with the EU regulatory framework.** WP3 dedicates effort to the implementation of an ETSI-compliant vNSF ecosystem and the management of lifecycle of its elements and services through the Store and the vNSFO.

Network function virtualisation (NFV) technology is one of the cornerstone technologies used within the SHIELD project and ETSI serves as one of its main standardisation drivers. The ETSI NFV architecture is used as the starting point for SHIELD's architecture, aiming to place SHIELD in a position where it can contribute with these standardisation activities and align itself to the de-facto industry standard. Thus, the software components envisioned in SHIELD's vNSF environment have been aligned wherever possible with the current vision of ETSI community. This vision/architecture may be extended as needed in order to accommodate components or features not yet considered or agreed by this standardisation body. The following figure (Figure 5) displays how SHIELD's architecture aligns with ETSI NFV architecture [1].

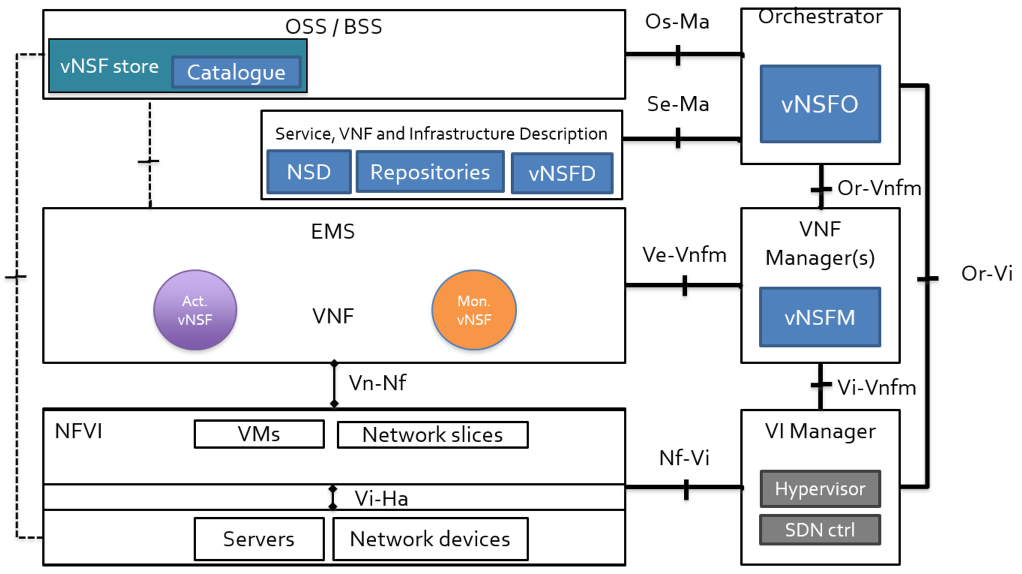


Figure 5: SHIELD vNSF environment's architecture mapped to ETSI NFV architecture.

The Store lies in **the Operational and Business Support layer**, whereas the vNSFO directly fits into the role of the **Orchestrator** envisioned in the ETSI NFV architecture and the vNSFs also have a direct mapping within the **VNF section**. The subcomponents and even modules or elements were successfully mapped as well, e.g. the NS and vNSF information (descriptors, records, infrastructure-related data, etc.), as well as the vNSF Manager (vNSFM) that directly corresponds to the VNF Manager following what ETSI envisions for the mechanism used to control the vNSFs (EMS subcomponent and so on). The only remaining component present in SHIELD's architecture, and in the scope of WP3, is the Trust Monitor; which performs attestation tasks that are not contemplated in the ETSI NFV architecture and thus the component has no direct mapping. **The attestation framework implemented in SHIELD is a necessity, since NFV adoption increases the infrastructure's attack surface thus unveiling potential cybersecurity concerns.**

The following subsections describe the design and architecture for the SHIELD's WP3 components, i.e. the list of vNSFs to be deployed in the network, the vNSF Store, the vNSF Orchestrator and the Trust Monitor. This description is more detailed than its counterpart in D2.2, as it specifically addresses low-level details such as the subcomponents within the vNSF environment, their detailed workflows and relation between these and other components in the SHIELD platform.

Regulatory compliance, especially in terms of the General Data Protection Regulation is also an important aspect in the design of the vNSF ecosystem. Compliance with the EU regulatory framework is essential, as it can enable or hinder adoption of the SHIELD platform. This aspect is further analysed in Section 4.

2.2. Security network functions and services

The NFV concept achieves, through virtualisation, the reduction of the capital expenditures incurred by common specialised hardware devices and provides a broad spectrum of network functionalities that are deployed on top of common hardware. The Network Services (NSs) and the Virtual Network Security Functions (vNSFs) they contain can be moved, restarted or erased rapidly, up to the order of seconds. vNSFs implement common network functions such as gateways, proxies, firewalls and transcoders, traditionally carried out by specialised hardware devices and deployed on top of commodity IT infrastructure. The focus within SHIELD is on the development of VNFs implementing security services and functions (hereinafter called NSs and vNSFs). To ease their management, the developed vNSFs will conform to the ETSI NFV group recommendations. The following subsections contain a general architecture to be followed by the SHIELD vNSFs.

2.2.1. General vNSF architecture

Each vNSF is composed by one or more VNF Components (VNFCs) that are interconnected through Virtual Network Links (VLs). The security services offered in SHIELD will consist of one or more vNSFs. These NSs will be dynamically deployed to identify and mitigate security attacks, threatening conditions or anomalous behaviours. The vNSFO will be responsible for the orchestration of the vNSFs into services and the deployment, management and configuration of the resulting end-to-end network services. An example of a network service (NS) that consists of three different vNSFs (VNF1, VNF2 and VNF3) connected through virtual links is shown in Figure 6. As depicted, VNF2 is composed by three VNFCs connected through virtual links that are internal to the VNF.

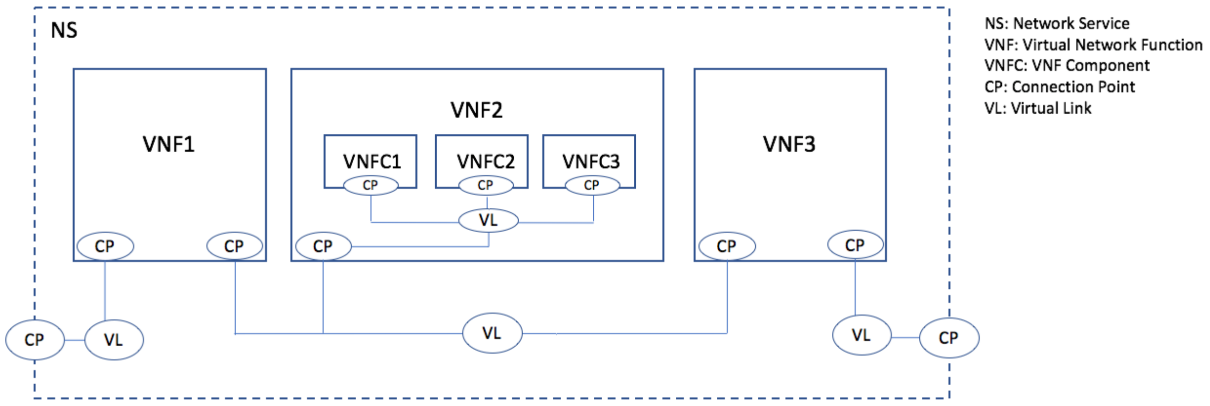


Figure 6: Network Service example.

2.2.1.1. vNSF interfaces

According to ETSI NFV specifications [1], there are five types of interfaces identified relevant to a VNF. As illustrated in Figure 7:

- SWA-1 interface: This interface enables communication between various network functions within the same or different network services. The SWA-1 interface can be established between two VNFs, a VNF and a Physical Network Function (PNF), or between a VNF and an End Point. A VNF may support more than one SWA-1 interface.
- SWA-2 interface: This interface refers to VNF internal interfaces, for the communication between the different VNFCs of a VNF, i.e. for VNFC to VNFC communication. The type of information exchanged through this interface depends on the function of the VNF.
- SWA-3 interface: This interface interconnects the VNF with the NFV management and orchestration layer specifically with the VNF Manager (VNFM). Through this interface the lifecycle management of the VNF is performed (e.g. instantiation, termination, scaling, etc.). The SWA-3 interface corresponds to the Ve-Vnfm reference point.
- SWA-4 interface: This interface is used by the Elemental Management (EM) to communicate with a VNF. It is a management interface used for the runtime management of the VNF to perform functions related to Fulfilment, Assurance, and Billing (FAB) as well as Fault, Configuration, Accounting, Performance and Security (FCAPS). This interface will cover also the NSF-facing interface's functionality defined in the IETF I2NSF standard, within the task for defining policy recommendations.
- SWA-5 interface: The SWA-5 interface links the VNF with the NFVI and corresponds to the Vn-Nf reference point. This interface provides access to a virtualised slice of the NFVI resources allocated to the VNF, i.e. to all the virtual compute, storage and network resources allocated to the VNF depending on the VNF type and its special requirements for resources.

As the SHIELD framework is compliant with the ETSI MANO specifications, the SHIELD vNSFs will support these interfaces.

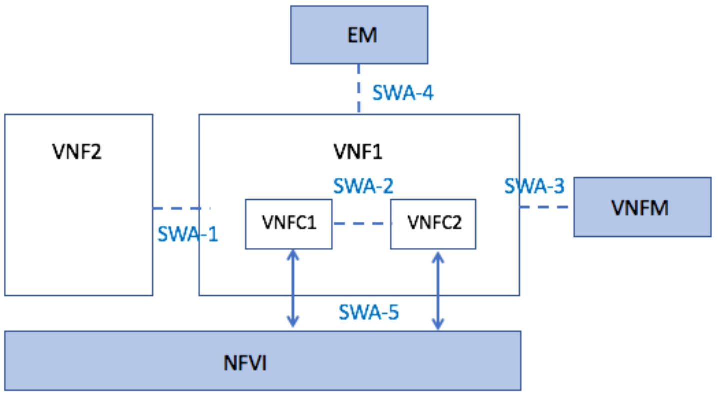


Figure 7: Types of VNF interfaces.

2.2.1.2. vNSF common elements

The internal structure of a SHIELD vNSF is illustrated in Figure 8. Although the internal implementation of a vNSF concerning its functionality (vNSF functionality) is to be decided by each vNSF developer, there are some common elements that vNSFs should have to be compatible with the SHIELD framework. Specifically, these elements are:

- The **vNSF controller** is the internal element devoted to the support of the vNSF lifecycle through the vNSFM. The interaction between the vNSFM and the vNSF takes place through the SWA-3 interface.
- The **init configuration** element is responsible for the initialisation of the vNSF that happens at the beginning of the vNSF execution. This is an optional component that is present when an initial configuration should take place on the vNSF before its execution.
- The **data collector** element is the component responsible for gathering the output data from the vNSF. The format of the output data follows a low-level application-dependent format.
- The **data transformation** element, whose role is to transform the output data of the vNSF from a low level, application-dependent format (Data Collector) to a high-level format that is understandable by DARE.
- The **configuration listener**, an element responsible to listen for new policy configurations recommended by the Remediation Engine of DARE and injected by the vNSFO into the vNSF.
- The **policy transformation** element, whose role is to transform the high-level format rules recommended by the Remediation Engine of DARE (policies) to low-level, application-dependent format rules that can be enforced to the vNSFs. This element will be part of each vNSF on which policy enforcement is expected to take place.
- The **streaming service** is the element responsible for transmitting application-level monitoring data, such as security logs or alerts produced by the vNSF, to the Streaming Service located at the DARE.
- The **vNSF Functionality** element represents the functionality performed by the vNSF.

It is important to note that the above elements are not what is typically referred to as vNFCs in ETSI terminology (or VNFCs in SHIELD). It is possible that all the above elements reside in a single VNFC. Additionally, apart from the vNSF Controller that allows the lifecycle management of each vNSF, none of the other described elements are mandatory for all vNSFs. The presence of the other components listed above is dependent of the type of each developed vNSF. The data collector, the data transformation and the streaming service will be present in all vNSFs that produce some output, which will be used by the Data Analytics Engine of DARE for the identification of security incidents and threats. Similarly, the policy transformation and the configuration listener will be present in all vNSFs that permit some application-level configuration for security purposes (threat identification or mitigation) through the Security Orchestrator.

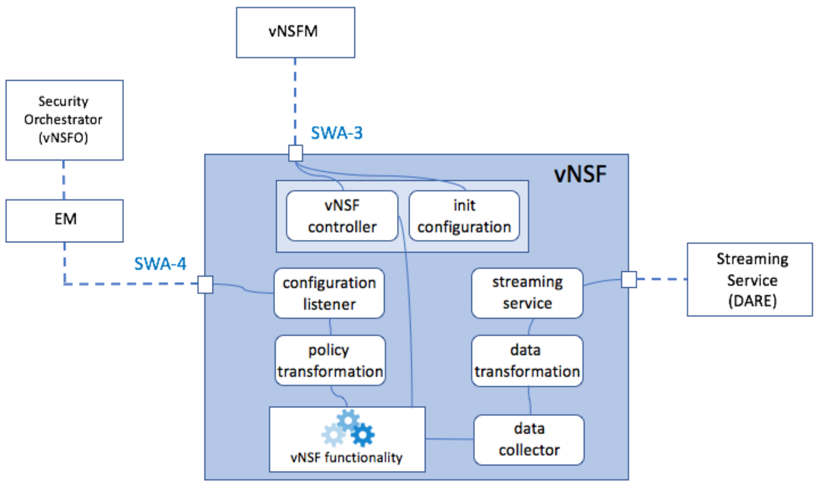


Figure 8: Internal structure of a SHIELD vNSF.

Figure 9 depicts the internal elements of a vNSF comprised of two VNFCs. In the case of having a vNSF comprised of more than one VNFCs, the vNSF Controller element will be present in one of the available VNFCs. Additionally, the vNSF Functionality element will be present in all VNFCs composing the vNSF. The remaining elements of the common vNSF architecture can be freely allocated in the different VNFCs, again taking into account the type and function of the vNSF (vNSF that provides output, vNSF that accepts configuration, etc.). In the specific example illustrated in Figure 9, the data collector, the LH data transformation and the streaming service components reside in the second VNFC (vNSF-C2).

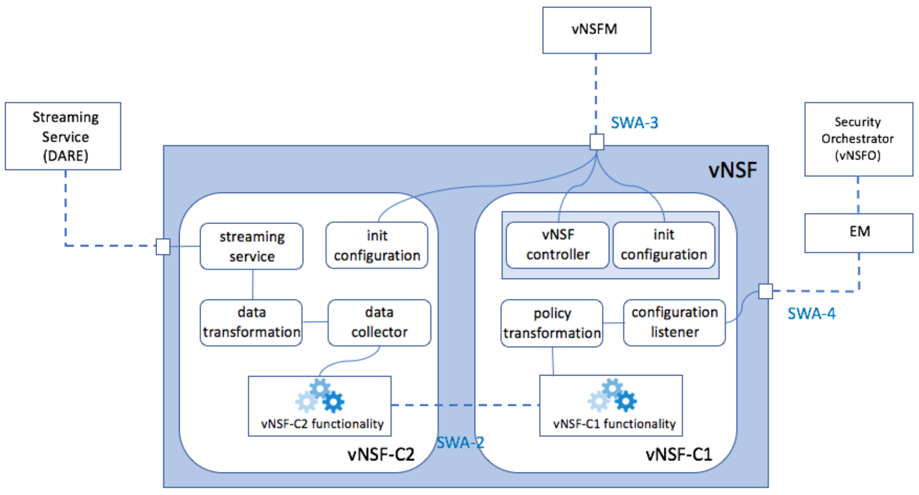


Figure 9: Internal elements of a vNSF composed by two VNFCs.

2.2.1.3. vNSF Descriptor (vNSFD) and NS Descriptor (NSD)

Each vNSF, as any VNF, has an associated descriptor document; whose role is to instruct the vNSFO on how to deploy and configure it, and how it should be connected to other virtual functions. This descriptor document, usually referred as VNF Descriptor (VNFD) is a deployment template which describes a VNF (or vNSF, in SHIELD) in terms of deployment and operational behaviour requirements. Information typically detailed in the VNFD contains deployment instructions, scaling policies, configuration information and monitoring parameters related to the function of the vNSF. Moreover, the VNFD will contain connectivity, interface and KPIs requirements that can be used by the vNSFO to establish appropriate virtual links between vNSF components instances, or between a vNSF instance and the endpoint interface to other virtual functions.

A similar descriptor file is associated with each network service (NSD), providing information and networking details on how the vNSFs connect to provide a given network service. Besides the VNFD and NSD, a vNSF or NS package in SHIELD will also provide some metadata via its security manifest. These metadata are intended to be used by attestation purposes, while avoiding to alter the constructs used for VNF and NS descriptor. The manifest contains hashes for different information and resources related to each package.

2.2.2. SHIELD NSs

SHIELD will implement several monitoring and remediation vNSFs. Monitoring security functions perform traffic monitoring and analysis to detect intrusions and report illegitimate traffic or malicious activity. On the contrary, the role of the remediation security functions is to mitigate security threats or risks by applying security policies and taking actions, such as dropping/rejecting specific packets or flows and blocking data coming from specific users. It must be noted that several vNSFs implemented in the project may assume both roles, i.e. monitoring and remediation.

2.2.2.1. Monitoring NSs

The monitoring vNSFs will probe the network in different ways to extract relevant low-level information from the NFVI. This network data is called “Network data collection” and its contents will vary depending on the purpose of each monitoring vNSF. After the network data collection is obtained, it is transformed from an application specific format into a high-level structure with a generic format via the “data transformation” process and then is sent to the DARE through the “Streaming Service” interface (as depicted in Figure 8). The rationale of converting the data to a generic format and provide the DARE’s Streaming Service with a generic format is to allow DARE’s compatibility with different implementations for a single vNSF type. For example, the definition of a generic format for monitoring data coming from intrusion detection systems would allow the compatibility with different IDS vNSFs implementations (e.g. Snort, Bro, Suricata etc.).

2.2.2.2. Remediation NSs

The reacting NSs will be in charge of providing mitigation actions, as defined by the DARE. The rules or policies composing a mitigation action —expressed via an application-independent configuration's abstraction— will be proxied to the vNSFO by the Security Dashboard, once accepted by the final user. Each reacting NS involved in a particular mitigation action will receive the set of policies via the SWA-4 interface (Figure 8) and will be in charge of translating it to low level configuration, understandable by the implemented security network function. Thus, the translation process will be offloaded to the different reacting NSs in a specific module, named policy transformation in Figure 8. This is done in order to reduce the load on the centralised points of the architecture, as well as to ease any modification/update in the translation process by the NS developer.

2.2.2.3. List of NSs

The consortium has selected a number of candidate NSs that will allow to demonstrate SHIELD capabilities (detection and mitigation) in security attacks. Specifically, the following NSs are targeted for implementation: An Intrusion Detection System (IDS), a mcTLS gateway, a traffic analysis NS, a deep packet inspection NS, a packet filter NS acting at network layer, a forward proxy NS acting at application layer. The specific functionalities selected for the NSs implementation depends on the security requirement analysis as defined in WP2 and on the security threats to be addressed during the project's demonstrations.

The detail associated with the specification of each NS differs based on its maturity level. As a consequence, some NSs already provide a detailed low-level specification of its internal architecture/workflow while others are still in a preliminary stage, therefore presenting only its envisioned functionalities.

Virtual Intrusion Detection System (vIDS)

An IDS is equipped with advanced traffic analysis and monitoring capabilities for attack and vulnerability detection. It monitors and logs the network traffic for signs of malicious activity and generates an alert upon discovery of a suspicious event. Two different techniques are used to detect malicious traffic/activity, separating IDSs into two main categories: i) statistical anomaly-based IDS and ii) signature-based IDS. Anomaly detection IDSs have the advantage over signature based IDSs in detecting novel attacks for which signatures do not exist. However, anomaly detection IDS suffer from high false detection rate. IDS deployment typically consists of one or more sensors placed strategically on the network. Additionally, the solution may contain an optional central console for easier management of all sensor nodes.

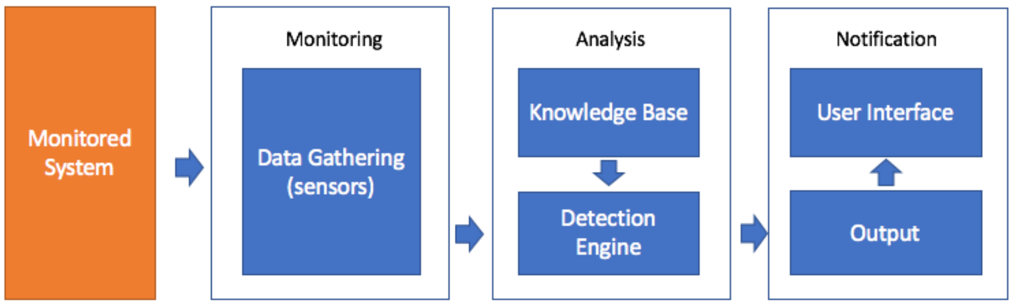


Figure 10: Typical IDS architecture.

The functionality of an IDS involves three distinct phases: a) Monitoring, b) Analysis and c) Notification. A typical architecture of an IDS is illustrated in Figure 10. During the Monitoring phase, the IDS is collecting data from the monitored system, through the deployed sensors. At the Analysis phase, the IDS Detection Engine analyses the gathered data by using a Knowledge Base. The Knowledge Base includes information that allows the Detection Engine to classify the analysed data as threatening events. This information includes predefined rules (signatures), user defined rules or historical data. The historical data allows the modelling of the normal behaviour of the monitored system into a profile enabling the detection of deviations of the current status when compared to this considered normal profile. Finally, during the notification phase, the IDS will output notifications of the detected events by logging this information into specific files and user interfaces or trigger alerts that can be consumed by other components. The following figure (Figure 11) illustrates the internal components of the IDS vNSF, which is comprised of a single VNFC.

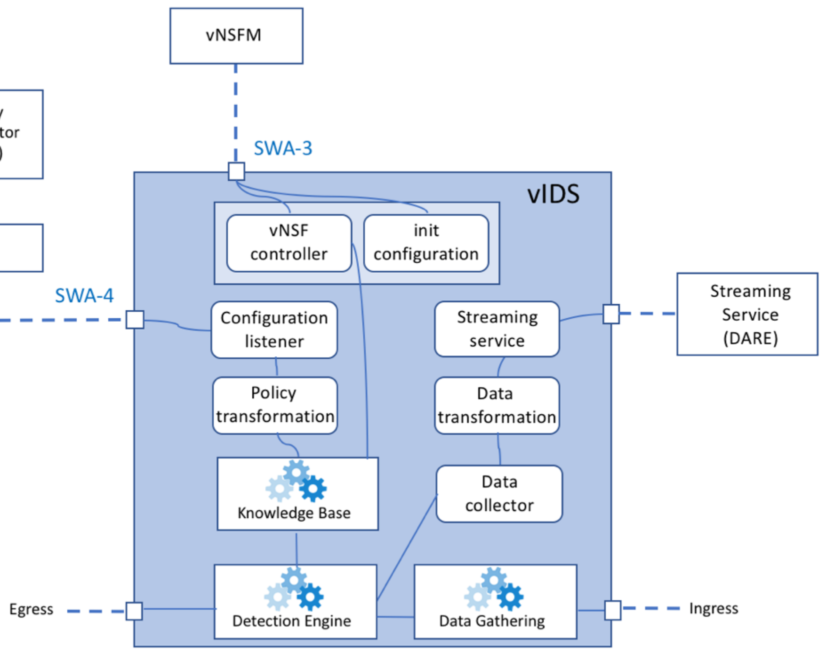


Figure 11: vIDS internal components.

Deep Packet Inspection (DPI)

DPI is the practice of filtering and examining IP packets, across Layers 2 through 7. Although Stateful Packet Inspection (SPI, often employed by firewalls) is more restricted, DPI may extend to headers, protocol structure and payloads, thus allowing for the implementation of advanced cybersecurity measures. DPI can be an effective detection tool for a multitude of cyberattacks such as Denial of Service (DoS), buffer overflow, cross-site scripting exploits, injection attacks etc. DPI capabilities, however, can be limited as the payload structure becomes more complex (e.g. through obfuscation, encryption etc). SHIELD aims to implement a vNSF dedicated to DPI, as part of the trusted platform. The scope of the SHIELD vDPI is to monitor cybersecurity events and log their evolution over time. The threat information and analytics can then be relayed to a CERT/CSIRT in using the Structured Threat Information Expression (STIX) information model.

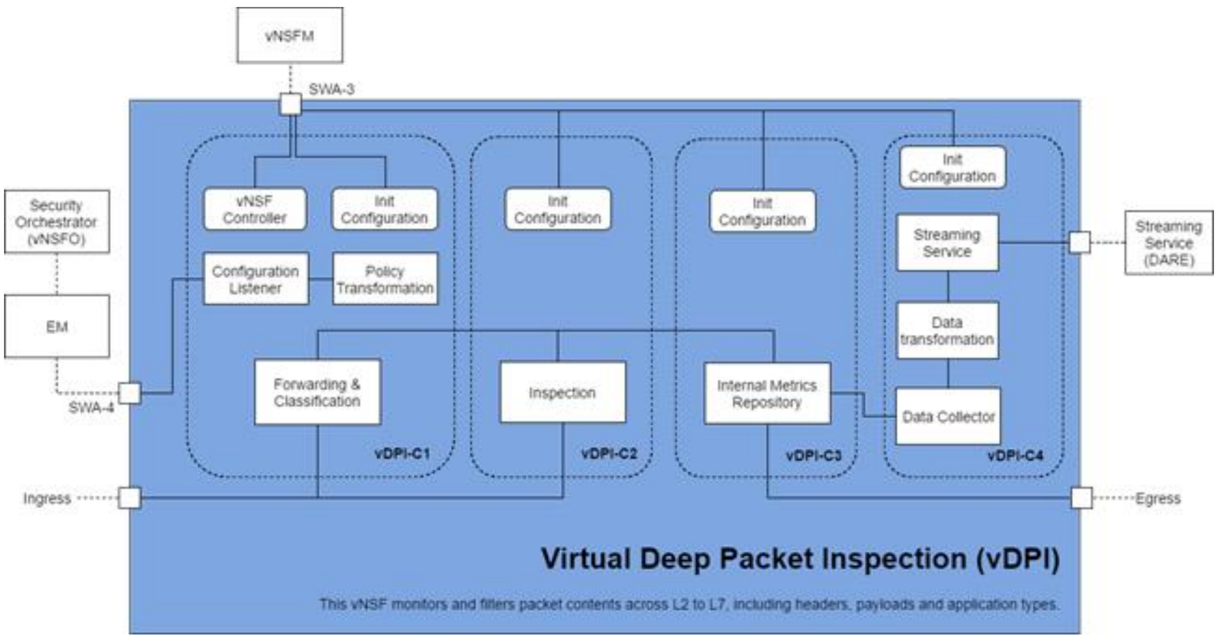


Figure 12: vDPI design and main components.

The vDPI NS encompasses a vDPI vNSF comprising of several vNSF components (VNFCs) as illustrated in Figure 12:

- **vDPI-C1 (Forwarding and Classification):** This vNFC handles routing and packet forwarding. It accepts incoming network traffic and consults the flow table for classification information for each incoming flow. Traffic is forwarded using default policies until it is properly classified, and alternate policies are enforced. It is often unnecessary to mirror packet flow in its entirety in order to achieve proper identification. Since a smaller number of packets may be utilized, the expected response delay can therefore be close to negligible. In a case where the Inspection, Forwarding and Classification VNFCs are not deployed on the same compute node, traffic mirroring may introduce additional overhead. A classified packet can be redirected, marked/tagged, blocked, rate limited, and reported to a reporting agent or monitoring/logging system within the network.
- **vDPI-C2 (Inspection):** The traffic inspection vNFC implements the filtering and packet matching algorithms and is the necessary basis to support additional forwarding and classification capabilities. It is a key component for the successful implementation of the vDPI and the most computationally intensive. The component includes a flow table and an inspection engine. The flow table utilises hashing algorithms for fast indexing of flows, while the inspection engine serves as the basis for traffic classification.
- **vDPI-C3 (Internal Metrics Repository) & vDPI-C4 (Monitoring Dashboard):** The internal metrics repository acts as local storage, while the Monitoring Dashboard handles data sharing with DARE.

The vDPI lifecycle is managed by the vNSFO, and specifically the vNSF Configuration Manager subcomponent. The vNSFO is in charge of starting, stopping, pausing, scaling and configuring the vDPI. Thus, the Forwarding and Classification component acts as a managing/controlling VNFC and is assigned a floating IP for management. Internal communication is implemented via vlinks (detailed in section "Specifications and Implementation"). Policies are relayed from the vNSFO and translated within the managing vNFC.

ProxyTLS

Original design for a Multi-Context TLS (mcTLS) [14] based in a secure protocol that extends TLS to incorporate trusted middleboxes into a secure session, has been modified because current state of this technology is not enough mature and widely accepted to be integrated in the SHIELD framework at this stage. On the contrary, the ACME STAR protocol [15] (Automated Certificate Management Environment / Short-Term Automatically-Renewed) is in a mature state and compatible with CAs in the market. STAR solution solves similar problem (TLS traffic inspection by trusted party) but from a different approach. STAR automatize the process of request and use temporal certificates from 3rd party domain owners. The proxyTLS solution integrate STAR as part of a Web proxy. The ProxyTLS allow to inspect HTTPS traffic with the aim of solve cybersecurity threats such as malicious URLs. One key difference in this proxy is the capability to inspect and log the complete URLs in the HTTPS header, in contrast to other security tool that can only see the TLD domain from the Certificate issued. The purpose of ProxyTLS is to monitor all HTTPS connection and log the URLs used by the clients. In addition, it can be populated with URL blacklists (i.e. malware droppers, C&C, phishing servers, etc.) to generate alerts and in the case of mitigation block the connectivity.

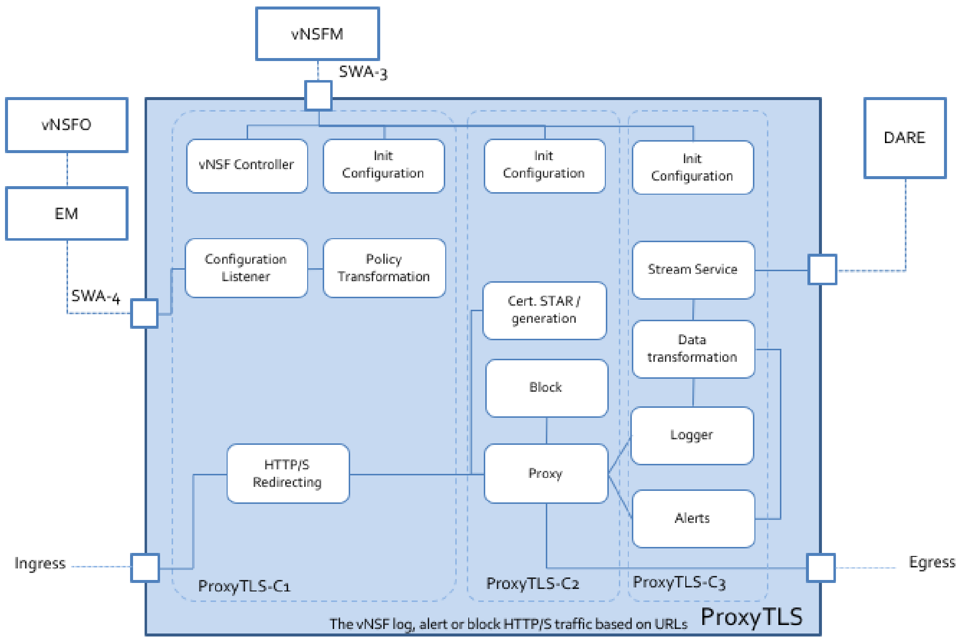


Figure 13: ProxyTLS Gateway elements.

As shown in Figure 13, proxyTLS will be deployed using several components:

- **ProxyTLS-C1 (HTTP/S redirecting)**: Provides the functionality to capture and redirect in transparent mode all HTTP and HTTPS traffic towards the proxy. It also includes all the management interactions with SHIELD.
- **ProxyTLS-C2 (Proxy)**: This component integrates as main functionality the HTTP/S proxy function. Ends existing TLS session and open a new against the server. In order to comply with browser security checks, impersonate on the fly the server certificate. For collaborative domain owners, implements the ACME STAR draft client to use valid server certificates. Finally, in case of activation the filtering capacity, the proxy blocks configured URLs and response with a notification HTTP static web.

- **ProxyTLS-C3 (DARE collector)**: The last component generates the vNSF activity, traffic logs, alerts and filtering actions and parse them to a compatible format with DARE. This information is sent to the DARE using a standard available stream service.

Functionalities provided:

- Middlebox for traffic monitoring: Inspect only HTTP and HTTPS header, including URLs information, User-agent and IPs.
- Potential direct interaction with other vNSFs able to block HTTP/S traffic when a security threat is detected through SHIELD framework.
- Simple HTTPS answer for blocked pages
- Support for monitor/block files with list of malicious URLs or domains,
- Short term temporal Certificate delegation and control from origin servers through the Automated Certificate Management Environment (ACME).

HTTP/S Analyser

The objective of this vNSF is to provide the classification of HTTP and HTTPS traffic without analysing the payload content in a privacy-friendly way. Increasing cybersecurity attacks are hiding behind HTTPS traffic, such as phishing attacks, targeted spam with web page links, data leak or botnet C&C channels. This vNSF offer the opportunity to generate a first classification of the HTTPS traffic, to discard or to make more direct security traffic inspection, by other tools, meanwhile keep the privacy of the communication, because avoid inspect traffic payloads. This vNSF will be trained through machine learning techniques to provide the HTTP traffic classification in order to be able to analyse the behaviour of a device or network. The vNSF will be able to work with the traffic mirror or with stored information in tstat [16] format.

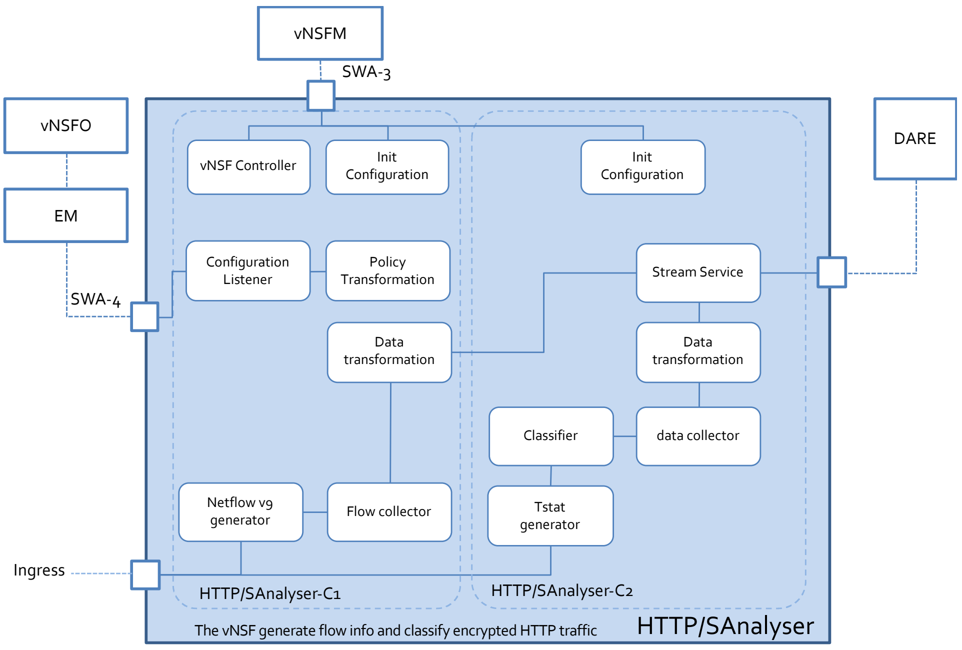


Figure 14: HTTP/S Analyser design and main components.

Figure 14 details the components of the HTTP/S Analyser:

- **HTTP/SAnalyser-C1 (Netflow Probe)**: A standard netflow v9 probe from mirrored traffic. Integrate also the collector component to send all the data to the DARE.

- **HTTP/S Analyser -C2 (Classifier)**: This component analyse the traffic and classify it with a level of confidence. Unknown flows are classified with another label.

Functionalities provided:

- Traffic capture, netflow and tstat format traffic generation.
- HTTP/S traffic and classification in several categories: BROWSING, VIDEO, STORAGE and OTHER by network flow. Traffic analysis is based only in OSI layer 2 to 4.

L3 Filter

This vNSF will implement a filtering application acting at the network layer, or Layer 3 of the ISO/OSI stack. It will allow or deny traffic by specifying an Access Control List (ACL), in form of a whitelist or blacklist. The ACL will be configured by translating the high-level configuration to a set of filtering rules for specific IP addresses, ports and transport protocol.

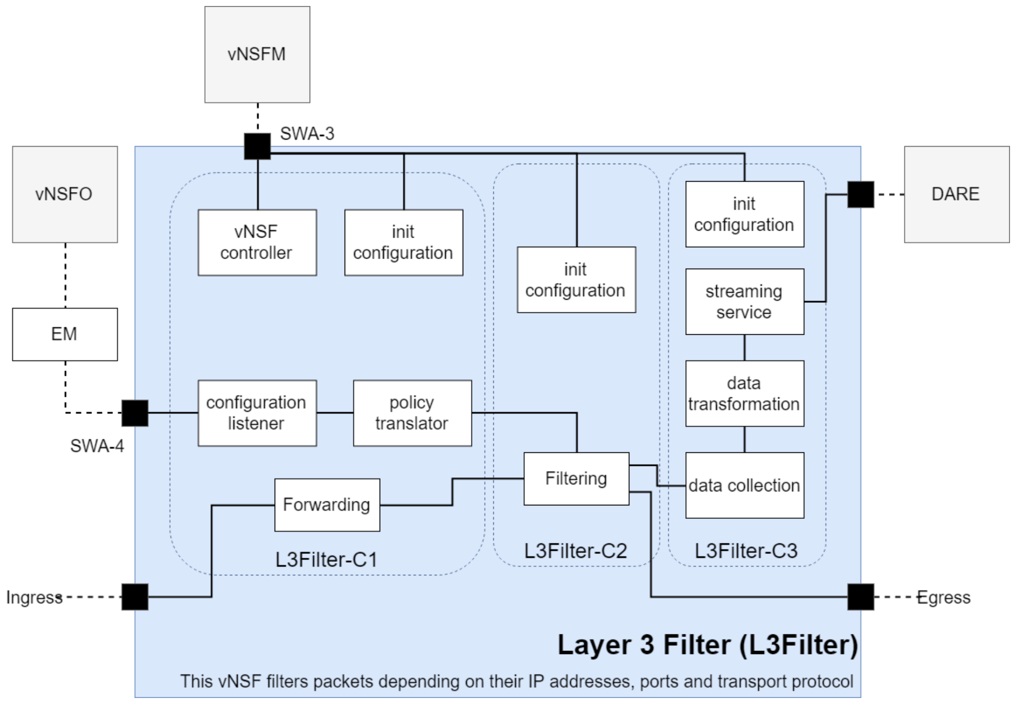


Figure 15: L3 Filter design and main components.

Functionality provided:

- Allow or deny traffic identified by a certain IP address (source, destination).
- Allow or deny traffic identified by a certain port (source, destination, protocol)
- Rate limit traffic

The L3 Filter is composed of different VNFC, as pictured in Figure 15:

- **L3Filter-C1 (Forwarding):** This VNFC handles packet forwarding for the ingress traffic, which will be redirected to the filtering engine. Moreover, this component acts as the vNSF controller, hence it includes a configuration listener to receive security policies from the vNSFO. These policies are translated in low level configuration by a local translator, and they are made available to the L3Filter-C2 component, which is in charge of applying them.

- **L3Filter-C2 (Filtering):** The Filtering VNFC filters the incoming traffic, according to its (source/destination) IP addresses, ports and transport protocol, leveraging the results of the policy translation. The filtering operation can result in dropping, allowing and rate limiting the incoming traffic, before forwarding it to the egress interface.
- **L3Filter-C3 (DARE Collector):** This component collects the application metrics of the Filtering VNFC and forwards them to the DARE for further analysis.

The L3 Filter shall be assigned a floating IP for management in its Forwarding component, which acts as the ingress component for both the traffic and the policy configurations. The management of this vNSF is performed by the VNF Configuration Manager subcomponent, within the vNSFO.

Forward L7 Filter

This vNSF will implement a forward proxy that would offer the possibility to block all the traffic the user wants to block. To do so, it will inspect traffic at application layer (also named Layer 7 in the ISO/OSI stack) and filter it according to defined rules. The vNSF will behave as an agent that will receive requests from a client (e.g. a web browser) and forward them to the specified server, if it doesn't match a blacklist. The Functionalities provided include:

- Traffic inspection for specific Layer 7 protocols and headers (e.g. HTTP, FTP);
- URL filtering;
- Access Control List (e.g. IP based, MAC based, domain based);
- Reverse Proxy.

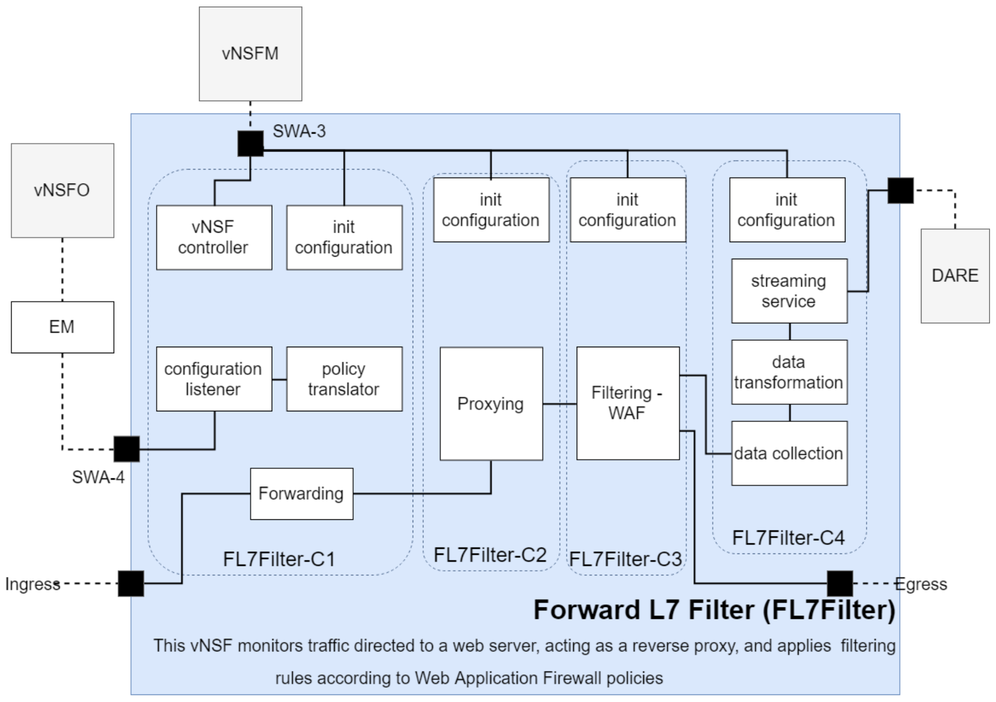


Figure 16: Forward L7 Filter design and main components.

The Forward L7 Filter is composed of different VNFC, as pictured in Figure 16:

- **FL7Filter-C1 (Forwarding):** This VNFC forwards traffic directed to one or more web servers to the Proxying component. It also acts as the vNSF controller and manages the lifecycle of the vNSF. Moreover, it includes a configuration listener and a policy translator in charge of configuring the virtual hosts for the reverse proxy and the filtering rules for the Web Application Firewall.
- **FL7Filter-C2 (Proxying):** This VNFC configures the virtual hosts for the web servers to be protected by the Forward L7 Filter vNSF. The virtual hosts shall not provide SSL configuration, as the key management is out of scope for this vNSF functionality. All the traffic gathered for each virtual host is forwarded by this component to the Filtering – WAF VNFC.
- **FL7Filter-C3 (Filtering - WAF):** This VNFC applies filtering rules for possible attacks targeted for Web Applications. It monitors the incoming traffic and classifies it according to rules that describe traditional application layer threats, such as SQL Injection and Cross Site Scripting. This VNFC includes an access log for the filtered traffic, which is made available to the DARE Collector VNFC. The allowed traffic is redirected to the egress interface of the vNSF.
- **FL7Filter-C4 (DARE Collector):** This component collects the application metrics of the Filtering - WAF VNFC and forwards them to the DARE for further analysis.

The Forward L7 Filter will require a floating IP to allow the management of its internal components. The vNSF Manager is the infrastructure component in charge of managing the lifecycle of the vNSF.

2.2.2.4. Functionality mapping

The following table describes, per vNSF, how these provide the specific monitoring or remediation capabilities.

Table 1: Functionality Mapping.

vNSF	Monitoring	Remediation
vIDS	Real-time traffic analysis (L3-L4 and L7) for intrusion detection based on signatures. It can also be used as a simple packet sniffer or packet logger. Cross-cutting all uses cases.	No
vDPI	Filtering and examining traffic (L2-L7), extending acquisition of headers, protocol structure, application types. Cross-cutting all uses cases, demo plans include Use case 3, exporting threat information to CERT/CSIRTs.	The vDPI does not apply remediation rules, although it can receive the offending IPs and monitor the evolution of a cyber incident. It also has the capability to export analytics and threat information to a CERT or CSIRT team. Targeting UC3.
ProxyTLS	Monitoring the Header payload of HTTP requests to identify threats. Cross-cutting all uses cases	Yes. Allow to filter provided URLs or preloaded list. Cross-cutting all uses cases

HTTP/S Analyser	Netflow probe and flow Classification of HTTP and HTTPS traffic using ML techniques, without analysing the payload content.	No
L3 Filter	No	Allow or deny traffic identified by a certain IP address (source, destination), port (source, destination) and transport protocol. Cross-cutting all uses cases
Forward L7 Filter	Analysis of Layer 7 headers and (optionally) payloads to classify possible Web Application-targeted security threats.	Filtering for specific Layer 7 protocols and headers (e.g. HTTP, FTP), URL filtering, Access Control List (e.g. IP based, MAC based, domain based)

2.3. Store

SHIELD aims to set up a single, centralised digital store for vNSFs and NSs. This approach allows SPs to offer new security features for protecting the network or extend already existing functionalities without the need of modifying core elements of the framework. The store acts as a repository for vNSFs and NSs that have been previously published. The main novelty in the Store is the onboarding of vNSFs/NSs in a secure and trusted way. The onboarding process will ensure the provenance is from a trusted source and that the contents integrity can be assured. The security information is then stored for safekeeping and provided upon request, so other components can check that the vNSF/NS has not been tampered with since it was onboarded.

Another relevant feature provided by the Store is the verification done on the vNSF and NS associated descriptors to ensure the instantiation process can be successfully performed. Building on the descriptors syntax check concept from the SONATA project [2], the submission process shall check all descriptors for inconsistencies as well as implement a network topology validation. This last check will prevent issues such as unwanted loops in the forwarding graphs or reference to undefined networks or missing ports. Figure 17 presents all the Store sub-components, along with their relations depicted.

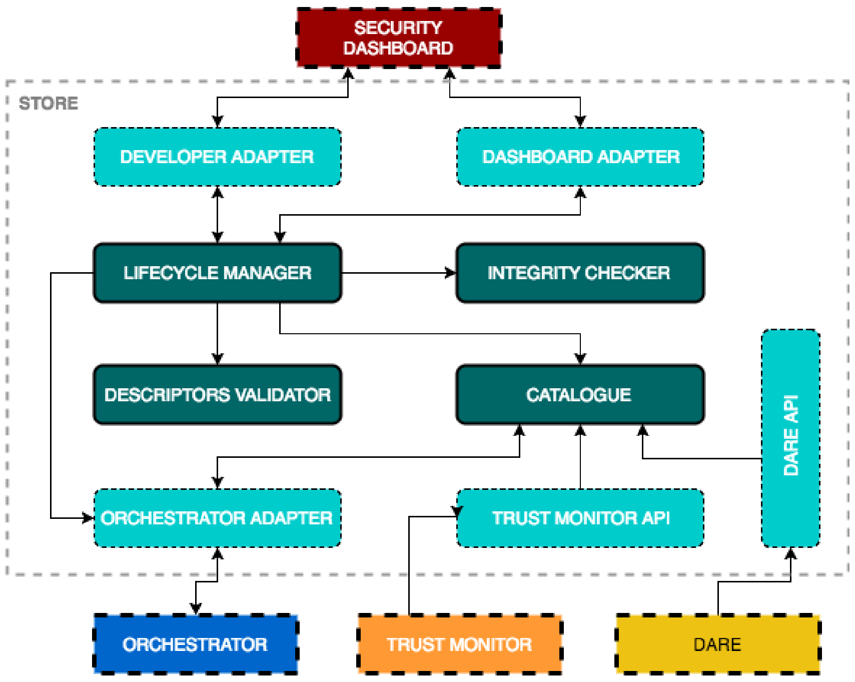


Figure 17 : vNSF Store subcomponents.

2.3.1. Subcomponents

The current section presents each subcomponent depicted in Figure 17 mentioning its main role. The STORE component encloses four main subcomponents (LIFECYCLE MANAGER, INTEGRITY CHECKER, DESCRIPTOR VALIDATOR and CATALOGUE) as well as four subcomponents aiming to provide connectivity with other SHIELD components. These subcomponents (DEVELOPER ADAPTER, DASHBOARD ADAPTER, ORCHESTRATOR ADAPTER, TRUST MONITOR API and DARE API) will be translated to either APIs (providing a connection point to external components); Connectors (using the features of external components); Adapters (enclosing both API and Connectors features:

- Lifecycle Manager: This subcomponent manages the vNSF/NS onboarding lifecycle. From the moment a NS/vNSF is submitted to the Store this sub-component takes over the entire process and ensures the proper steps are performed for a successful onboarding. In the event of a failure it notifies the Developer of the situation and performs all the necessary housekeeping steps.
- Descriptors Validator: Successful vNSF/NS onboarding consists of parsing its descriptor and validating the specified deployment and operational behaviour requirements. This job is performed by the Descriptors Parser sub-component. The two main tasks assigned to this component are syntax validation to prevent incorrect vNSF/NS descriptors from being processed for instantiation, and topology validation to assure the integrity of the vNSF/NS topology and avoid inconsistencies such as potential loops in the forwarding graphs or referenced to an undefined network or missing ports.
- Integrity Checker: When submitting a vNSF/NS to the Store the Developer must provide a manifest of the files used (or referenced) by the vNSF/NS. This manifest must contain hashes of each referenced file and must be digitally-signed so its contents can be trusted. It is paramount to a secure environment to ensure that the vNSF/NS content is trusted and wasn't tampered with in any way once on-boarded. The goal of the Integrity

Checker sub-component is to verify the integrity and provenance of the submitted data. This process encompasses validating the manifest which holds the hashes for the all files, as well as the ones regarding the descriptors. This information is provided later on to any component assessing that the vNSF/NS wasn't tampered with.

- Catalogue: All the onboarded and sandboxed vNSFs/NS are kept in a repository. The Catalogue sub-component manages the records-keeping activities. Any additional metadata associated with the onboarding process or the vNSF/NS itself is managed here as well.
- Developer Adapter: It provides the Developer with a REST API for onboard and withdrawal of vNSF and NS.
- Dashboard Adapter: It provides the Security Dashboard with an API for vNSF and NS catalogue management, as well as billing information.
- Orchestrator Adapter:It holds a connector to the Orchestrator, so the Store can onboard and withdraw vNSF and NS (using the Orchestrator API) and provides an API to the Orchestrator so it can query for vNSF and NS descriptors.
- Trust Monitor API: It provides the Trust Monitor with an API to query for vNSF security-related information which it uses to determine whether a vNSF has been tampered with.
- DARE API: It provides the DARE with an API to query for onboarded vNSF and NS.

2.3.2. General workflow

The Store interacts with multiple components, both in the vNSF environment (vNSFO, Trust Monitor) to share NSs and vNSFs data available in the catalogue; it also connects with other components of the SHIELD platform (DARE and Security Dashboard) for analytics and visualisation purposes. Besides this, the Store exposes endpoints to the NS/vNSF developers to onboard new NSs. The data flow diagram of the Store (Figure 18) depicts these interactions.

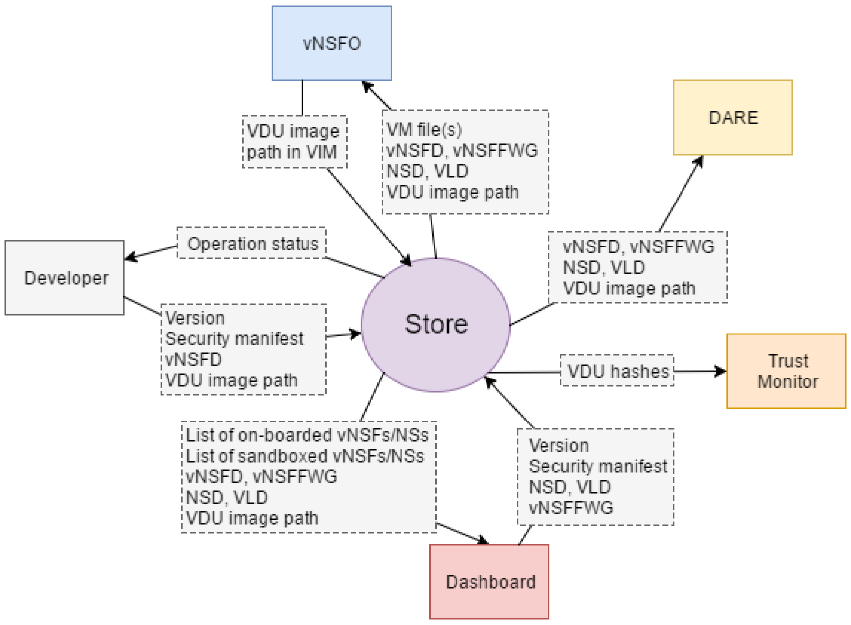

vNSFO
VDU image path in VIM
VM file(s) vNSFD, vNSFFWG NSD, VLD VDU image path
DARE
Operation status
Developer
vNSFD, vNSFFWG NSD, VLD VDU image path
Store
Version Security manifest vNSFD VDU image path
VDU hashes
Trust Monitor
List of on-boarded vNSFs/NSs List of sandboxed vNSFs/NSs vNSFD, vNSFFWG NSD, VLD VDU image path
Version Security manifest NSD, VLD vNSFFWG
Dashboard

Figure 18: Data flow diagram of Store.

2.3.3. Internal operation

The subcomponents of the Store work together to perform operations related to the onboarding process, such as the validation of the vNSF and the registration of the VDU image(s); as well as the decommissioning of NSs and vNSFs. The specific workflows for such operations are described in the "Annex A: Intra-component interactions".

2.3.4. Interactions with other components

The Store interacts with other components, namely the vNSFO, the Trust Monitor, the DARE and the Security Dashboard; as well as with the end users. Specific details are provided in the "Annex B: Inter-component interactions".

2.4. Orchestrator

The vNSFO (the vNSF Orchestrator used in SHIELD) builds on an implementation of an NFVO implementation that is ETSI NFV MANO (Network Functions Virtualisation Management and Orchestration) compliant. Additional features are implemented as part of the SHIELD vNSFO in order to support attestation and mitigation-related operations; as well as providing more convenient interaction between the SHIELD components. Orchestration refers to the deployment of the NSs (made up of vNSFs) and the management of their lifecycle, while also performing the global resource management, monitoring, configuration, translation and proxying of the VIM and NFVI resource requests, etc. Building on the feature of the orchestrator that enables the configuration of the vNSFs and NSs, the vNSFO receives medium-level security policies (MSPL); which are a sort of mid-level configurations that are produced within the DARE and are requested to be applied on a specific running vNSF. The vNSFO provides an interface to pass such configuration to a specific type of vNSF.

Finally, the attestation functionality requires the vNSFO to be more aware of the NFVI where it runs. To do so, it registers information on physical nodes and tracks extra information from virtual nodes. This data is provided to the Trust Monitor when registering new nodes to attest its integrity. The specific functionality is delegated to specific subcomponents and modules (Figure 19).

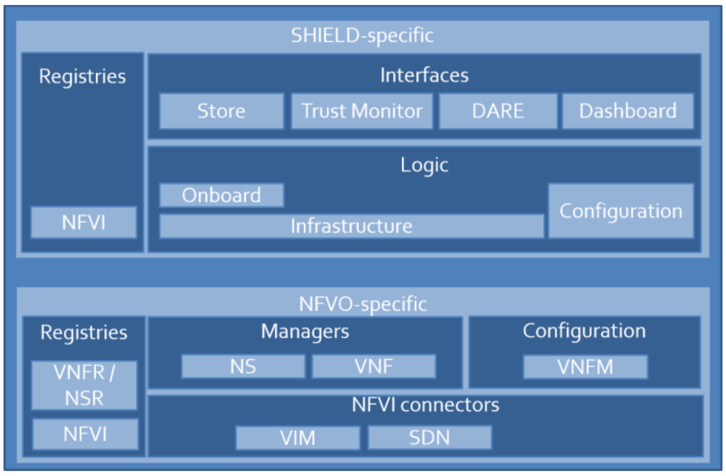


Figure 19: vNSFO subcomponents and modules.

2.4.1. Subcomponents

The functionality of the vNSFO is distributed among the NFVO-specific and the SHIELD-specific.

The former group comprises the operation of an NFV orchestration (through the NS and VNF managers, the VNF Manager's configuration and the NFVI connectors for one or more VIMs and SDN controllers). The latter introduces new additions that supports specific features introduced by SHIELD; namely the mitigation of attacks through the configuration of the SHIELDS vNSFs via MSPL and the periodic attestation of the physical and virtual infrastructure.

These are explained below:

- NFVO-specific
    - **NS and VNF Manager:** These managers control the lifecycle of any given NS or VNF. These can instantiate (deploy) or terminate (destroy) a given service or function into/from the NFVI. Other capabilities, like monitoring and scaling, are supported by the NFVO.
    - **Configuration VNF Manager:** It enables the modification of the configuration for the VNF during runtime. In SHIELD, the vNSFO receives a configuration request through the SHIELD-specific interfaces, which identify a type of vNSF or a specific running instance. Later, the vNSFO proxies the higher-level sort-of-configuration (MSPL) to the specific running vNSF.
    - **NFVI Connectors:** a subset of connectors and/or plug-ins to enable communication between the NFVO with the different components of the managed infrastructure. This is the mean to request and manage any physical or virtual-backed resource.
    - **Registries:** Data is persisted for operational purposes. The vNSF and NS Managers, the Configuration VNF Manager and the NFVI connectors all persist and fetch data to define records for run-time information on the operation of the deployed NSs/vNSFs, on the management information of the VIM and managers in the NFVI. Such data is required for multiple operations: from attestation to analytics or mitigation, and also for visualisation purposes.
- SHIELD-specific
    - **Interfaces:** The vNSFO exposes data through APIs and implements connectors to consume other components' APIs:
        - **Store API:** During the onboarding process, a developer uploads a specific NS and vNSF(s) through the Store. At the end of this process, the Store requests the onboarding of both packages into the vNSFO. This effectively registers them into the NFVO instance and make these available for later use.
        - **Dashboard API:** A user may select a recommendation from the Security Dashboard in order to deploy a proposed type of NS in the network infrastructure and mitigate a given threat. Then, the vNSFO is requested to inject the Medium-level Security Policy Language (MSPL) policies into the vNSF(s), which ultimately allows configuring the specific vNSFs.
        - **Dashboard Connector:** The vNSFO provides the Dashboard with data on the NFVI and running instances for visualisation purposes; namely the network topology and the running instances, according to specific filters.

- Trust Monitor Connector: The vNSFO contacts the Trust Monitor in order to register any newcomer node to the NFVI. With this data, the TM periodically performs the attestation on any given virtual or physical nodes. In case the attestation fails, the node shall be excluded from the NFVI.
- Trust Monitor API: The vNSFO provides the Trust Monitor with information on the network, the flow tables and the list of active nodes.
- DARE API: The vNSFO provides the DARE component with the topology of the network, the list of instances (according to specific filters) and the active deployed instances.

- Logic
    - Onboard: The onboard operation, in SHIELD, stems from the Store; which performs part of the process and delegates to the vNSFO the final stage (the onboarding into the NFVO).
    - Infrastructure: Interactions with the NFVI that are in charge of retrieving infrastructure-related information.
    - Configuration: The vNSFO can communicate to any specific vNSF so that any specific request middle-level policies are conveyed there.

2.4.2. General workflow

The vNSFO communicates with other components in the vNSF environment to receive vNSF and NS-related information during onboarding (Store), support the attestation of the security state of the running vNSFs and receive notifications during specific attestation stages (Trust Monitor), as well as inject policies that are later translated into usable configuration (vNSFs). The orchestrator communicates as well with other components in the SHIELD platform in order to receive policies for the vNSFs (Security Dashboard) and to provide up-to-date status on the network and vNSF status (DARE). The data flow diagram of the vNSFO (Figure 20) depicts these interactions.

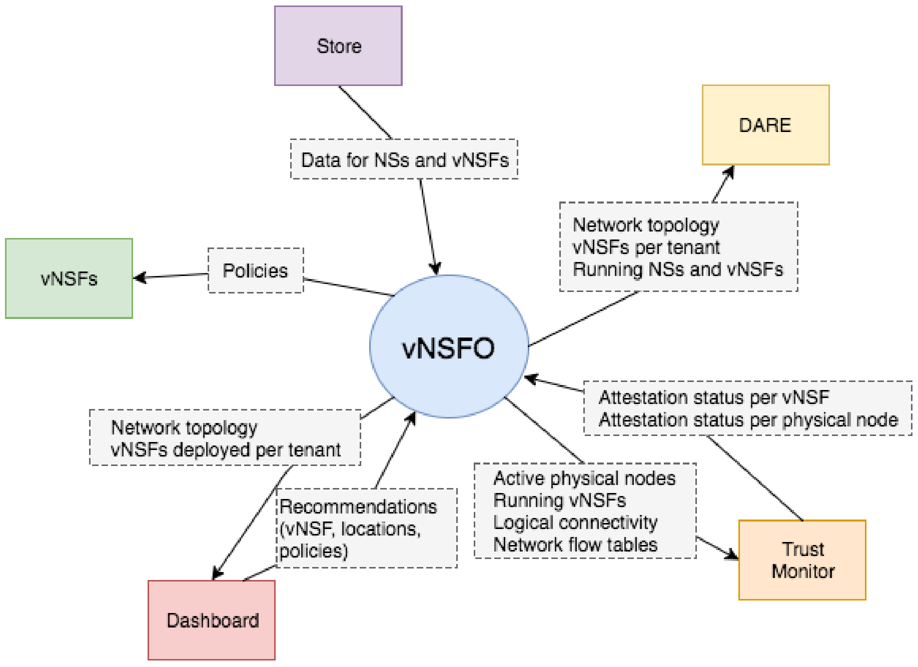


Figure 20: Data flow diagram of vNSFO.

2.4.3. Internal operation

From the SHIELD-specific operation, the exposed interfaces and related logic do interact with the NS and vNSF Managers. These work closely to manage the lifecycle of the NSs and vNSFs. Other subcomponents complement the operation of such instances with the ability to communication actions to the vNSF(s) and inject policies to these. The operations are described in the “Annex A: Intra-component interactions”.

2.4.4. Interactions with other components

The vNSFO interacts with the Store, NFVI, Trust Monitor, DARE and Security Dashboard in order to obtain information on NSs, deploy their resources, attest them and gather information to support analytics and visualisation. Specific details are provided in the “Annex B: Inter-component interactions”.

2.4.5. Comparison with I2NSF

The I2NSF (Interface to Network Security Functions) IETF working groups defines a set of software interfaces and data models that, among others like monitoring and querying, it allows inserting rules in the security-targeted NS (Figure 21). Such security NS are called Network Security Functions, or NSFs, in I2NSF and Network Services, or NSs (implicitly consisting of one or more vNSFs), in SHIELD.

```
                  +------------------------+
                  |    I2NSF Controller    |
                  +------------------------+
                            ^        ^
                            |        |
               +-----------+        +------------+
               |                                 |
               v                                 v
+ - - - - - - - - - - - - - - - +  + - - - - - - - - - - - - - - - +
|  NSF-A  +--------------+      |  |  NSF-B  +--------------+      |
|         | NSF Manager  |      |  |         | NSF Manager  |      |
|         +--------------+      |  |         +--------------+      |
| + - - - - - - - - - - - - - + |  | + - - - - - - - - - - - - - + |
| |+---------+    +---------+| |  | |+---------+    +---------+| |
| || NSF-A#1 | ... | NSF-A#n || |  | || NSF-B#1 | ... | NSF-B#m || |
| |+---------+    +---------+| |  | |+---------+    +---------+| |
| |         NSF-A cluster    | |  | |         NSF-B cluster    | |
| + - - - - - - - - - - - - - + |  | + - - - - - - - - - - - - - + |
+ - - - - - - - - - - - - - - - +  + - - - - - - - - - - - - - - - +
```

Figure 21: vNSF instantiation management in I2NSF framework.

The I2NSF framework (RFC 8329) [17] describes the “Security Controller” as the entity in charge of translating the high-level requests received through the Customer-facing Interface into actual control actions on the NSFs. Such control actions are related to the function semantics, and therefore deemed to be handled by the Entity Manager (EM) in the ETSI NFV framework.

In SHIELD, the role of the Security Controller is played by the vNSFO (since it encompasses the NFVO, which provides EM capabilities), as well as by the specific subcomponents in the Trust Monitor (generating the high-level policies) and in the DARE (translating the high-level policies to medium-level policies). Altogether, these provide the functionality to define the security policies and configure the vNSFs with them. In the SHIELD project, this final translation between policies and specific configuration is a process mostly carried within the vNSF itself.

2.5. Trust Monitor

The Trust Monitor (TM) assesses the trust in the network infrastructure bearing the deployed vNSFs, namely the NFVI Points of Presence (PoP) and the hardware network devices (e.g. switches). The trustworthiness of the infrastructure is assessed by performing both authentication and integrity verification.

Although attackers tend to exploit multiple vectors to breach into a system, the Trust Monitor focuses on intrusion detection in the network infrastructure, assuming the control and management plane components (vNSF store, vNSFO, DARE, Security Dashboard) are implicitly trusted. From a technical standpoint, extending the TM security concepts to assess the control and management plane, is feasible since they are based on the same kind of computer architecture (in terms of operating system, virtualisation technology and application packaging). SHIELD's threat model considers the following threats, classified on whether the attacker has physical access to the infrastructure or not:

- Physical threats:
  - T1 - physical eavesdropping: on network wire, bus probing;
  - T2 - physical modification of nodes: chip replacement;
  - T3 - physical introduction of a new/alternate control plan;
  - T4 - flashing of firmware/software of the network infrastructure nodes (e.g. in a network switch);
- Software threats:
  - T5 - zero-day vulnerability exploitation;
  - T6 - malicious (or accidental) administration: configuration modification, crafting SDN rules update;
  - T7 - installation and execution of arbitrary firmware/software (e.g. in the vNSF level);

SHIELD aims at providing the network infrastructure with detection mechanisms against software-based and low-end physical attacks: T1 and T2 are clearly out-of-scope since SHIELD does not provide any physical perimeter protection. Using Trusted Platform Module (TPM), remote attestation and other Trusted Computing mechanisms, the TM protects SHIELD's network infrastructure against T3, T4, T6 and T7. Particularly, the TPM protected log of all binaries executed on a node allows the TM to detect arbitrary code (T4 and T7). The same mechanism can be used to detect unwanted configuration modification (T6). If an attacker manages to introduce a new control plane entity in the network infrastructure (T3), the TM does not detect it directly but instead would detect any unusual or modified behaviour of the computer or network nodes since it would not be correct compared to the genuine control plane components, mainly the vNSFO. The TM verifies each node against their expected state, as configured by the vNSFO; if an attacker introduces a new control plane entity and changes – even slightly – the configuration of one node, the TM will detect it since it will not match the vNSFO's view. Looking at T5, this cannot be detected by the TM or regular Trusted Computing mechanisms. Nevertheless, zero-day vulnerability can be reduced by using code analysis tools and/or prevent their consequences by reducing the ability of the attackers. Mechanisms such as control-flow protection for instance, could help with that task. Even though, these kinds of attacks are usually the initial attack vector to install additional malicious software on the target, the execution log, verified by the TM, permit to detect the subversion.

Each physical node must be successfully authenticated - using hardware-based cryptographic identities - and verified by the Trust Monitor before joining the SHIELD infrastructure. Leveraging the Remote Attestation workflow, as defined by the Trusted Computing (TC) [18] paradigm (see "Annex C Definition of technologies"), the TM can verify the integrity of the code being executed (e.g., running instances of vNSFs, software directly managing virtualisation processes, etc) on each physical node, as well as its configuration, both at boot and run-time. The TM acts as a continual verification engine for the physical infrastructure hosting the NSs, capable of interacting with the rest of the vNSF ecosystem (vNSFO, vNSF Store) as well as the DARE to provide an assessment of the trustworthiness of the infrastructure.

Each NFVI node, being equipped with a TPM and suitable software, is able to collect the integrity measurements of both running code (starting from boot-time) and configuration, it is also able to report this data to a third party in a secure and trusted way. The resulting integrity report, which contains the logged software events - as measured by the Integrity Measurement Architecture (IMA) [7] for example - is validated by the Trust Monitor, which maintains a whitelist populated by measurements of known software signatures and their valid configurations. Network-related configuration, including the dynamic Software-Defined Network forwarding rules, is verified by the Trust Monitor as well, using the overall view available in the vNSFO. Trust Monitor subcomponents are identified in Figure 22.

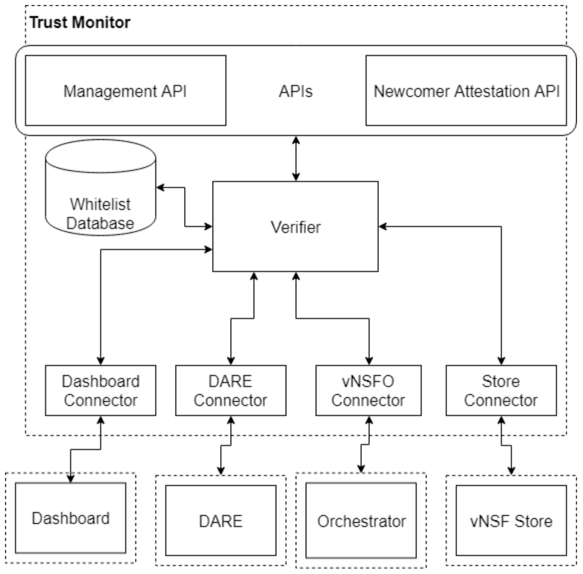


Figure 22: Trust Monitor subcomponents.

2.5.1. Subcomponents

A description of the TM's subcomponents depicted in Figure 22 is provided below:

- **Verifier:** It performs the TC-compliant Remote Attestation operations on each component that has been pre-registered with it. It performs both initial attestation of newcomers, periodic attestation tasks and notification of security events to both the DARE and the vNSFO. Each target must run specific software to gather the integrity measurements and send back this information to the Verifier.

- Whitelist Database It contains the list of measurements of known software - for both the platform and the vNSFs - and valid configuration. The list of known measurements for each vNSF is gathered from its security manifest in the vNSF Store. It should be noted that vNSFs are versioned in the Store, which allows detection of vNSF tampering (and hence the need to update the Whitelist Database) and the simultaneous use of different versions of the same vNSF.
- vNSF Store Connector: This connector is used to receive requests for integrity information from the store for each vNSF to be attested. This subcomponent is responsible for querying the vNSF Store via a client API and for retrieving the data required for the attestation of the vNSF: code in execution, with a special emphasis on custom applications that are not available from the standard software repositories, and configuration files required by the integrated security function, at deployment and runtime. This information is required to keep the Whitelist Database up to date with the measurements of the software components needed for the execution of the vNSF. The TM updates the Whitelist Database only when it detects that a new vNSF, or an updated version of it, is deployed in the NFVI.
- DARE Connector: This connector sends security events to the DARE if one physical or virtual instance is detected as compromised by periodic attestation, or in case a newcomer fails during authentication or initial integrity validation. The subcomponent's workflow is triggered by the Verifier.
- Dashboard Connector: This connector sends events on the infrastructure trust status to the Dashboard, in order to notify the end-user. The subcomponent's workflow is triggered by the Verifier along with the DARE Connector.
- vNSFO Connector: Through the vNSFO connector, the TM notifies the vNSFO about the need to terminate a compromised vNSF or to exclude a physical node from the NFVI. This workflow is triggered by the Verifier upon a failed attestation. In addition, it is used to request the configuration of the network at a given time from the vNSFO. The configuration consists of the description of active physical nodes, running virtual instances, logical connectivity and network flow tables.
- Newcomer Attestation API: It exposes an API that receives requests from the vNSFO for remote attestation of a node of the NFVI. The attested node must be pre-registered with the TM before performing the attestation procedure.
- Management API: This is a read-only interface for retrieving infrastructure attestation status data.

2.5.2. General workflow

The purpose of the TM is to assess the trustworthiness of the nodes composing the NFVI, in order to act on compromised nodes (e.g. exclusion from the NFVI) and validate the integrity state of newcomers. To do so, the TM should be able to interact and cooperate with several other components of the SHIELD infrastructure, such as the vNSFO, vNSF Store, etc. An overall description of the flows between the TM and the other component of the infrastructure is depicted in Figure 23.

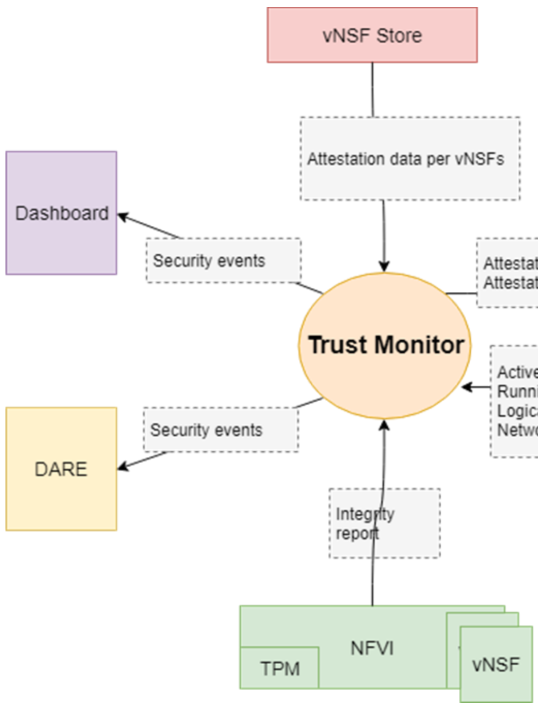


Figure 23: Data flow diagram of Trust Monitor.

2.5.3. Internal operation

To assess the trust of the NFVI (both physical nodes and virtual instances), the Trust Monitor needs to keep an updated list of known measurements about software packages and valid configurations. To do so, it interacts with the vNSF Store to retrieve the information needed for performing attestation of vNSFs, packaged within the security manifest of each network function's instance. Additionally, the Trust Monitor can download and measure packages of various Linux distributions from the official repositories and can also keep internal knowledge of the software updates for each of them. This data is used by the TM to attest the infrastructure nodes and rate them with different trust levels (e.g. by considering untrusted a node with a known software vulnerability).

The Trust Monitor is also able to keep an updated view of the network infrastructure at a given time by a specific interaction with the vNSFO, which in turn updates the Trust Monitor with status changes of the NFVI. This information can then be utilised by the Trust Monitor to periodically attest the NFVI and detect any compromised node. In addition, the vNSFO could directly ask the Trust Monitor to attest a node joining the NFVI, referred as "newcomer". The whitelist of known measurements can be used for checking the integrity report provided by each physical node of the NFVI during the Remote Attestation workflow. If any of the verification steps fail, the Trust Monitor notifies the vNSFO and logs the event in the DARE.

2.5.4. Interaction with other components

The Trust Monitor interacts with the Store, vNSFO, DARE and Dashboard components of the SHIELD's infrastructure to request attestation-related information or as a response of an external attestation request. A detailed description of each workflow is presented in the "Annex B Inter-component interactions".

3. Specifications and Implementation

The information conveyed in this section decreases the abstraction level for the software solution provided. Based on the components and sub-components defined in the architecture section it presents additional insight on the inner details of said sub-components by defining implementation-oriented behaviours, operations and interactions. Such behaviours may be supported by software design elements such as data flows, state machines, decision flows or API/interfaces descriptions.

Targeting an implementation-oriented approach this section references possible technologies or features from existing technologies to use, reused outcomes or extensions to develop based on other projects or even specify features to create from scratch. To assist the reader in understanding how the selected technologies fits within SHIELD rationale, the requirements fulfilment is also included.

3.1. Security network functions and services

This section describes the vNSFs identified so far to perform monitoring and remediation within the scope of the SHIELD platform. For each of them a mapping of its functionality against a subset of the SHIELD requirements is provided, as well as low level specification and implementation details when available.

3.1.1. Virtual Intrusion Detection System (vIDS)

3.1.1.1. Implementation details

For the implementation of a virtualised Intrusion Detection System in SHIELD it is planned to adopt the IDS VNSF [19] that was developed in the frame of CHARISMA project. Several modifications and extensions will be made to support full compatibility with the SHIELD platform. The vIDS vNSF, as used in CHARISMA, includes the following components:

- **Snort IDS**: An open-source intrusion detection system, capable of performing real-time traffic analysis and packet logging on IP networks [20].
- **Barnyard2**: An open-source software tool that takes Snort output and writes it to a SQL database to reduce load on the system [21].
- **PulledPork**: An open-source tool that automatically downloads the latest Snort rules (threat signatures) [22].
- **Snorby**: An open-source web-based graphical interface for viewing and clearing events logged by Snort [23].
- **Rule Configuration Service:** A service that accepts requests for creating, deleting and modifying rules that can be applied in Snort detection engine.
- **Event Publisher Service**: A service responsible for publishing the alerts produced by Snort detection engine.

The current CHARISMA IDS vNSF implementation is based on Ubuntu 14.04 operating system, which was selected as the guest operating system in CHARISMA project. Incoming traffic to the

IDS vNSF is being analysed in real time and analysis decisions are being communicated to external interfaces as HTTP requests. This vNSF consists of one virtual machine which requires to have one virtual network interface where all traffic that need to be monitored must be routed (or mirrored). Thus, the CHARISMA vIDS contains a single VNFC. Additionally, the vNSF is accompanied by an ETSI compliant descriptor that allowed its life-cycle management through the TeNOR (T-NOVA) orchestrator.

The IDS implementation is based on Snort open source IDS. Snort [20] is an open-source intrusion detection system that is developed by Sourcefire. It can perform real-time traffic analysis and packet logging on IP networks. Snort architecture is composed by the packet capture library, the packet decoder, the pre-processor, the Snort detection engine which is configured with detection rules and the alert output components plug-ins.

Rule Configuration Service

To provide intrusion detection functionalities based on policy defined by external modules to the vIDS, this VNSF implements a RESTful API which accepts requests for creating, deleting and modifying rules that can be applied in Snort detection engine. This offers an easy way of external configuration of the VNSF without requiring knowledge of its inner workings.

Event Publisher Service

The IDS VNSF provides another functionality, necessary for the utilisation of the results produced by Snort packet analysis, the Event Publisher Service. This service translates, curates, and publishes events in readable format to external interfaces for further analysis. Once traffic enters the IDS vNSF, Snort software analyses all packets. Snort detection engine, described above, can contain rules which consist of conditions. When the conditions of a rule are met, the detection engine produces an event and saves it in a log file. Snort event logs are saved in Unified2 format, so the Event Publisher Service translates them to JSON format, assesses their timestamp to avoid publishing redundant information and publishes the events.

A number of modifications to the CHARISMA IDS vNSF to make it compatible with the SHIELD platform are foreseen. More specifically:

- Virtualisation enabler: A CentOS 7.X will be used as the guest operating system to provide a virtual machine-based IDS for SHIELD. Additionally, a second version of the IDS will be provided bundled in a Docker container or -if required- multiple Docker containers.
- vNSF descriptor: The vNSF descriptor of the vIDS will have to be implemented from scratch to allow life-cycle management through the OSM orchestrator.
- Rule Configuration Service: This component matches the configuration listener element included in all vNSFs that accept configuration through the Security orchestrator. Modifications to the current implementation are expected to allow compatibility with the Security Orchestrator and the exact format of the policies sent.
- Event Publisher Service: This component matches the streaming service element included in all vNSFs that provide monitoring information data to the DARE. Modifications to the current implementation are expected to allow compatibility with the data format expected from the DARE Streaming Service.
- User interface and output: As SHIELD platform features a User Dashboard for displaying output and threat alerting and notifications to the user, it is unlikely that the Snorby GUI component will be required for the SHIELD vIDS implementation.

3.1.1.2. Requirements mapping

Requirement	Requirement name	Requirement description
SF08	DoS Protection	A security service SHALL protect against volumetric Denial of Service attacks. Detect the DoS attack and divert the traffic for filtering. Forwarding the good traffic flows to the destination.
VI_SPEC_01	vIDS will perform traffic analysis against its signatures database to detect a DoS attack and notify DARE about it; which will in turn instruct specific mitigation procedures.	
SF09	Intrusion Detection/Prevention System	A security service SHALL detect attacks with a wide range of techniques such as network flow or behaviour analysis and deep packet inspection. Allow traffic flows according to IPS rules. Monitor traffic network traffic at OSI layer 7 and generate alerts for security policy violations, infections, information leakage, configuration errors and unauthorised clients.
VI_SPEC_02	vIDS will analyse the traffic in L3-L4 and L7, generating appropriate alerts upon any detected intrusion and notify DARE regarding identified security threats or incidents. After internal analysis and correlation, DARE will instruct specific mitigation procedures.	
NF05	Impact on perceived performance	When network traffic is proxied or analysed, the user experience SHALL not be degraded.
VI_SPEC_03	The traffic analysis carried out by the IDS should not seriously delay or degrade the detection and mitigation operations.	
Resource Requirements	1 CPU, 50GB disk storage and 2GB RAM are the minimum requirements for the IDS vNSF. 1 CPU, 50GB disk storage and 4GB RAM is the recommended setup.	

3.1.2. Virtual Deep Packet Inspection (vDPI)

3.1.2.1. Implementation details

The implementation of the vDPI components is based on a variety of technologies allowing to perform traffic inspection as well as packet capturing. The following technologies are currently envisioned to be used in the implementation of this vNSF:

- **nDPI** [24]: is an open source alternative to the OpenDPI [25] library, maintained by ntop. Its goal is to extend the original library and add new protocols that are otherwise available only on the paid version of OpenDPI. Furthermore, nDPI is modified to be more suitable for traffic monitoring applications, by optimising the DPI engine. One of its major advantages is that nDPI can support application-layer detection of protocols, regardless of the port being used.

- **PF_RING** [26]: is a set of library drivers and kernel modules, which enable high-throughput packet capture and sampling. The PF_RING kernel module library polls packets through the Linux NAPI. Packets are copied from the kernel to the PF_RING buffer for analysis with the nDPI library.
- **DPDK (Data Plane Development Kit)** [27]: comprises of a set of libraries that support efficient implementations of network functions through access to the system's network interface card (NIC). DPDK offers to network function developers a set of tools to build high speed data plane applications. DPDK operates in polling mode for packet processing, instead of the default interrupt mode. The polling mode operation adopts the busy-wait technique, continuously checking for state changes in the network interface and libraries for packet manipulation across different cores.

The PF_RING implementation selected for the vDPI has the capacity of maintaining uninterrupted connectivity with the OpenStack network. DPDK has the capacity to bypass the Linux kernel, leading to high-performance packet capture but is less robust and fault-tolerant that PF_RING.

3.1.2.2. Requirements mapping

Requirement	Requirement name	Requirement description
PF05	Analytics visualisation	The operator SHALL be able to see the analytics visualised in e.g. a dashboard.
VD_SPEC_01	vDPI will be able to monitor specified flows and show how a cyber incident evolves over time. The analytics will be shown in a graphical user interface along with exported statistics in STIX 2.0 [28] format.	
SF08	DoS Protection	A security service SHALL protect against volumetric Denial of Service attacks. Detect the DoS attack and divert the traffic for filtering. Forwarding the good traffic flows to the destination.
VD_SPEC_02	vDPI will be able to monitor suspicious traffic and report attack statistics and observations in a graphical user interface. vDPI will export threat information in STIX 2.0 format, showing the observed evolution of the attack and communicating the results to an appropriate cybersecurity agency or incident response team. vDPI will not be performing behavioural analysis and will not be looking into the contents of a packet but rather the headers for the selected suspicious flows.	
PF12	Threat data sharing	Sharing threat data with a third entity SHALL be allowed. The granularity of such data depends on the severity and type of each attack.
VD_SPEC_03	The vDPI will be able to export threat information in STIX 2.0 format. It will monitor identified suspicious flows over a period of time.	
PF17	Interoperability	The platform SHALL expose openly-defined APIs for information exchange with third parties.

VD_SPEC_04	The selection of the STIX 2.0 format for information exchange was made to satisfy monitoring and reporting requirements posed by cybersecurity agencies. The operator may select to allow access to STIX information for CERT/CSIRT, threat databases, security/data protection audits etc.	
SF09	Intrusion Detection/Prevention System	A security service SHALL detect attacks with a wide range of techniques such as network flow or behaviour analysis and deep packet inspection. Allow traffic flows according to IPS rules. Monitor traffic network traffic at OSI layer 7 and generate alerts for security policy violations, infections, information leakage, configuration errors and unauthorised clients.
VD_SPEC_05	vDPI will offer deep packet inspection capabilities based on the nDPI library. Capabilities include inspection of packet headers, applications types etc., but not deep content inspection (which requires reassembly and inspection of an entire message). vDPI is intended as a monitoring component and will not be imposing rules to block/limit traffic although it will export statistics and threat information.	
NF05	Impact on perceived performance	When network traffic is proxied or analysed, the user experience SHALL not be degraded.
VD_SPEC_06	The traffic inspection performed by vDPI should not seriously degrade the user's quality of experience on the NS. vDPI engine will be based on open source high-throughput tools (nDPI, PF_RING etc) and will be able to parse small subsets of mirrored traffic.	
Resource requirements	2 virtual processors, deployed in 40GB storage/4GB memory flavour is recommended.	

3.1.3. ProxyTLS

3.1.3.1. Implementation details

This vNSF is based on the combination of several open source solutions to offer a holistic security functionality. All the solution runs over a Linux distribution, and make use of the iptables Linux kernel module to implement the HTTP traffic redirection towards the proxy. The additional components are based on:

- **Mitmproxy** [29]: This open source project implements a versatile HTTPS proxy. Between their functionalities we can mention: Intercept HTTP & HTTPS requests and responses and modify them on the fly, save complete HTTP conversations for later replay and analysis, replay the client-side of an HTTP conversations, replay HTTP responses of a previously recorded server, reverse proxy mode to forward traffic to a specified server, transparent proxy mode on OSX and Linux, make scripted changes to HTTP traffic using Python or SSL/TLS certificates for interception generated on the fly. The ProxyTLS function use some of them, such the modify the certification generation on the fly, modify the HTTP response and create customized logs compatible with DARE collector format.

- **CertBot** [30] **and Boulder** [31]**:** This two tools represents the client and Server for Automatic Certificate Management Environment (ACME). CertBot is a command line tools that allows to request periodically to a CA to issue of signed certificates for its owns domains in an automatic way. The Boulder is the CA software to automatize the delivery process. This code has been developed and used by Let's encrypt [32], who it is the responsible for a clear increment in the use of HTTPS in Internet in the recent years. For the proxyTLS a modified version of the Boulder and the CertBot has been made to align the code with ACME STAR draft, to allow temporal certificates issues to 3rd parties.

3.1.3.2. Requirements mapping

Requirement	Requirement name	Requirement description
SF02	Detect/Block access to malicious network locations	A security service SHALL control access to malicious network locations, such as phishing servers, malware spreading websites, Command & Control (C&C) servers, etc. The user must be alerted and the access to the site could be blocked/allowed depending on the configured policy rule.
VP_SPEC_01	The vNSF (middlebox) allows the monitoring of HTTP ciphered traffic directed at any server (in order to identify attacks). The ProxyTLS will inspect the HTTP headers. It will compare the URLs from the headers against black lists to detect any of the malicious endpoints.	
NF04	Impact on perceived performance	When network traffic is proxied or analysed, the user experience SHALL not be degraded.
VP_SPEC_02	When network traffic is proxied or analysed, the user experience SHALL not be perceived as degraded.	
Resource Requirements	Recommended requirements are 4 virtual processors, deployed in 20GB storage and 4GB memory. Minimum is 1 processors, 10GB storage and 1GB RAM.	

3.1.4. HTTP/S Analyser

3.1.4.1. Implementation details

This vNSF implementation is based on Linux standard distribution for the operative system. Additionally, both components implement different open source tools, for traffic capture and aggregation. Dataset generation, processing and flow classification are done by new developed application in the classifier. This module imports pre-trained model for specific classification. This classifier is based in the results from the COGNET project, and it is being deployed as a standalone VNF.

Next are the list of tools involved:

Nfdump [33]: This software suite allows to capture network traffic, create and store netflow v9 flows. It allows also inject in the network the netflow protocol. This software is used to generate all the traffic received on a mirror port

Tstat [16]: This solution generates traffic flows summary and group them by type of categories. One of the main properties of this tool is that generate multiple information features for each network flow, very suitable for Machine Learning technology.

3.1.4.2. Requirements mapping

Requirement	Requirement name	Requirement description
SF09	Intrusion Detection/Prevention System	A security service SHALL detect attacks with a wide range of techniques such as network flow or behaviour analysis and deep packet inspection. Allow traffic flows according to IPS rules. Monitor traffic network traffic at OSI layer 7 and generate alerts for security policy violations, infections, information leakage, configuration errors and unauthorized clients.
VH_SPEC_01	Traffic classification will allow the classification of encrypted traffic traversing the network and therefore enable its correlation with other security sources to detect attacks therefore improving its detection/mitigation mechanisms.	
PF04	Security data monitoring and analytics	The platform SHALL be able to collect and analyse Events from the vNSFs in real time in order to detect security incidents
VH_SPEC_02	Traffic events generation can be configured. HTTP/S Analyser will generate and share netflow and tstat flow events to share with DARE.	
NF07	Compliance to standards	The platform SHALL conform to well-established standards, in particular with respect to data export (e.g. STIX) and input (e.g. NetFlow).
VH_SPEC_03	Not only for HTTP but for any type of traffic this VNF can generate NetFlow v9 standard format.	
Resource Requirements	Recommended requirements are 4 virtual processors, deployed in 100GB storage and 8GB memory. Minimum is 2 processors, 10GB storage and 2GB RAM.	

3.1.5. L3 Filter

3.1.5.1. Implementation details

The implementation of this vNSF will be based on the packet filtering framework included within the Linux kernel, starting from the 2.4 version. The framework, maintained by the netfilter.org project, consists of different subsystems, such as iptables [34]. This userspace program can be used to configure the filtering ruleset, composed of rules consisting of classifiers (e.g. the source IP address) and one connected action (e.g. deny).

The vNSF will provide an Access Control List in a standard format, such as XML, containing a list of IP addresses to be allowed or denied, depending on the kind of list (whitelist, blacklist). The vNSF will manage the low-level translation of the ACL to iptables rules.

The implementation of this vNSF will be based on one of the following technologies:

- Iptables, subsystem of the netfilter framework included in the Linux kernel (since version 2.4);
- Pf [35], a firewall distributed with BSD license and integrated in OpenBSD and FreeBSD.

Both the technologies support basic filtering according to Layer 3 and 4 headers, including IP addresses, ports and protocol. Moreover, they support a rate limiting capability for both source IP addresses and a target IP address. The two implementations differ from the performance perspective, as Pf leverages a stateful tracking functionality to enhance the performance at the increase of filtering rules. Because of this, Pf would be the most advanced solution to implement the vNSF and additional tests are being performed to ensure FreeBSD support for vNSF in the Open Source MANO framework.

An implementation for a L3 packet filter, based on iptables, has been developed in the scope of the SECURED project [4], and will be considered as base point for development of this vNSF.

3.1.5.2. Requirements mapping

Requirement	Requirement name	Requirement description
SF04	L4 traffic filtering	A security service SHALL monitor traffic based on configuration rules. Traffic packets are filtering, and specific traffic is either allowed, rejected or blocked based on a predefined set of rules (usually based on source IP, destination IP, destination port, etc.).
VL3_SPEC_01	The L3 Filter vNSF filters the incoming traffic depending on IP address, port, transport protocol.	
SF06	Malware detection	A security service COULD detect (and optionally clean) files with malware downloaded from Internet.
VL3_SPEC_02	The L3 Filter vNSF can be configured to log connections initiated by a host to non-internal IPs. These logs can be further analysed to understand if any host is	

	executing unauthorized connections to an external destination, such as a malware source.	
SF08	DoS protection	A security service SHALL protect against volumetric Denial of Service attacks.
VL3_SPEC_03	The L3 Filter vNSF can filter the IP addresses or ranges that are used by the DoS attackers to mitigate a volumetric DoS threat. The vNSF can also apply rate limiting rules to the traffic.	
SF09	Intrusion Detection/Prevention System	A security service SHALL detect attacks with a wide range of techniques such as network flow or behaviour analysis and deep packet inspection. Allow traffic flows according to IPS rules. Monitor traffic network traffic at OSI layer 7 and generate alerts for security policy violations, infections, information leakage, configuration errors and unauthorised clients.
VL3_SPEC_04	The L3 Filter vNSF is expected to be integrated with the IDS/DPI as a remediation vNSF, blocking any malicious traffic detected by the IDS.	
NF05	Impact on perceived performance	When network traffic is proxied or analysed, the user experience SHALL not be degraded.
VL3_SPEC_05	The filtering operation performed by the vNSF should not seriously degrade the user's quality of experience on the NS.	
Resource Requirements	The minimum requirements of the L3 Filter vNSF include a CPU with 2 virtual cores, 10GB of storage and 4GB of RAM. The recommended requirements include a CPU with 4 virtual cores, 50 GB of storage and 8 GB of RAM.	

3.1.6. Forward L7 Filter

3.1.6.1. Implementation details

This vNSF will be implemented by leveraging the interaction of two web application-oriented technologies, namely Apache HTTP Server (HTTPD) and ModSecurity [36]. HTTPD [37] is an open source technology that implements a Web Server. Its mod_proxy extension makes it usable as a reverse proxy, acting as a gateway between the user agent (e.g. the browser) and the web server. ModSecurity is an open source Web Application Firewall (WAF), a toolkit that manages access control, monitoring and real time logging for web applications hosted by web servers. It is also named HTTP intrusion detecton tool, as it can detect threats with a rule-based approach. In fact, ModSecurity has a negative monitoring model, where a transaction, if not blocked by a specific rule, is always allowed. Each transaction can be analysed in distinct phases, which must be performed in order:

1. Request Header
2. Request Body
3. Response Header
4. Response Body

5. Log

ModSecurity has built-in support for the “Reverse Proxy” mode, where it can be automatically attached to the Apache HTTP Server. Moreover, ModSecurity can be integrated with the set of rules defined by the *Open Web Application Security Project* (OWASP), named OWASP Mod Security CRS [38], including triggers for code injection, broken authentication and session management, cross-site scripting.

Several other technologies have been analysed to select the proper technology for the vNSF. Apart from HTTPD, Apache Traffic Server [39], HAProxy [40], Varnish Cache [41] and nginx [42] have been analysed and considered valid for the reverse proxy functionality. Regarding the WAF functionality, the NAXSI project [43] has been evaluated as well. This proposes a positive monitoring model, meaning that the WAF allows the forwarding of the traffic to the web server by default. Moreover, NAXSI implements a whitelist approach that requires a classification of the traffic patterns in order to understand if certain packets are allowed or not. While NAXSI is supported by nginx only, ModSecurity is supported by nginx and HTTPD as well, and it performs better with the latter (in terms of throughput). Because of the positive model and the lack of a rule-based approach (which best suits a dynamic deployment of such security network function), the final choice was to implement a solution based on HTTPD and Mod Security.

3.1.6.2. Requirements mapping

Requirement	Requirement name	Requirement description
SF09	Intrusion detection/Prevention system	A security service SHALL detect attacks with a wide range of techniques such as network flow or behaviour analysis and deep packet inspection.
VL7_SPEC_01	The Forward L7 Filter vNSF will be able of monitoring and blocking L7 traffic depending on different criteria in order to control access to malicious websites (such as by filtering HTTP data according to blacklists of URLs). The traffic inspection will be performed against a set of rules as defined by the OWASP ModSecurity CRS.	
NF05	Impact on perceived performance	When network traffic is proxied or analysed, the user experience SHALL not be degraded.
VL7_SPEC_02	The filtering operation performed by the vNSF should not seriously degrade the user’s quality of experience on the NS.	
Resource Requirements	At minimum, the Forward L7 Filter vNSFs requires a CPU with 2 virtual cores, 20 GB of storage and 4GB of RAM. The recommended requirements include a CPU with 4 virtual cores, 100 GB of storage and 8 GB of RAM.	

3.2. Store

Based on the general architecture of the Store component provided in previous sections, the present section aims providing a preliminary specification of its low-level functionalities.

3.2.1. Specifications

For each subcomponent of the Store component, the low-level specifications are provided below.

Lifecycle Manager

The Lifecycle Manager subcomponent is responsible for implementing a set of features that enable the envisioned onboarding lifecycle of either vNSFs and NSs. The vNSF/NS onboarding lifecycle comprises the following steps:

- **Submission:** A vNSF has been submitted to the Store for onboarding by a Developer. Due to the nature of the process, as it comprises time-consuming operations such as validations and considerable-sized downloads, the submission request is promptly acknowledged, and the process continues in the background. Later, the Developer will be notified whether the operation succeeded or failed.
- **Sandboxing:** A vNSF is registered in the Catalogue but is not yet ready for production. It is undergoing a validation process to determine whether it is deemed fit for service.
- **Onboarding:** A vNSF has successfully undergone all the required checks to be considered able to integrate the ecosystem and is fit for attestation tests.
- **Decommissioning**: A vNSF has been taken out of service and can no longer be instantiated.

Descriptors Validator

To ensure a vNSF/NS can be onboarded, the descriptors provided in the package need to be validated. These descriptors are checked for:

- Syntax errors to prevent incorrect vNSF descriptors from being processed.
- vNSF topology integrity to avoid potential loops or errors such as references to undefined network interfaces.

Every onboarded vNSF descriptor will be checked for syntax, correctness and completeness issues. With no issues found the next step is to check the defined network topology and ensure inconsistencies such as no unconnected interfaces are present, and all virtual links are properly defined. Upon successful validation, the vNSF may proceed with the onboarding process. Any error results in a notification to the Developer stating what is not compliant with the SHIELD requirements. As for Network Services, onboarding the descriptors provided in the package need to be validated. These descriptors are checked for:

- Syntax errors to prevent incorrect NS descriptors from being processed.
- vNSF/NS topology integrity to avoid potential loops or errors such as references to undefined network interfaces
- Decommissioned vNSF usage to avoid service instantiation issues

Again, every NS descriptor will be checked for syntax, correctness and completeness issues. With no issues found the next step is to check whether any usage of decommissioned vNSF is present. Upon successful validation, the NS may proceed with the onboarding process. Any error results in a notification to the Security Dashboard stating what isn’t compliant with the SHIELD requirements.

Integrity Checker

The vNSF onboarding security check is performed by:

- Verifying the package digital signature against the stored one to prove provenance.
- Checking the hashes for the vNSF-related files against the ones provided in the manifest to ensure integrity.

The security manifest format is defined by SHIELD and all submitted vNSFs, regardless of intended target vNSFO, shall comply with it (no tailoring is allowed). Upon successful checks the vNSF may proceed with the onboarding process. Any error results in a notification to the Developer stating what is not compliant with the SHIELD requirements.

Catalogue

The Catalogue handles the records for the entire Store component. It stores data of all the onboarded vNSFs and NSs and can convey it to the other components upon request through the adapters provided for such purpose. The specific data is defined below:

- **vNSF Catalogue:**
  - Version: an identifier for the submitted vNSF package which defines a unique set of specific functionalities and dependencies provided within the vNSF-related descriptors.
  - Status: the current state of the vNSF. It can be “submitted”, “sandboxed”, ”onboarded” or “decommissioned”.
  - Security manifest: holds the hashes for all the vNSF-related files as well as information needed for attestation.
  - vNSF Descriptor (vNSFD): description for the vNSF, containing the VNFCs that conform the vNSF, the available flavours to deploy and the description of the virtual links interconnecting the different VNFCs.
- **NS Catalogue:**
  - Version: an identifier for the submitted NS package which defines a unique set of specific functionalities and dependencies provided within the NS-related descriptors.
  - Status: the current status of the NS be it submitted, sandboxed, onboarded or decommissioned.
  - Security manifest: holds the hashes for all the NS-related files as well as information needed for attestation.
  - NS Descriptor (NSD): description for the service, containing the vNSFs that conform the service and their forwarding graphs, the virtual link description interconnecting the vNSFs, the preferred flavour (instance configuration) per vNSF to use and any SLA to be met by the NS.
  - Virtual Link Descriptor (vLD): definition of the virtual network links that interconnect the vNSFs.
  - vNSF Forwarding Graph Descriptor (vNSFFGD): definition of the network deployment for the vNSFs contained in the NS.
- **Developer Adapter:** This module provides connectivity with the Developer either in the form of an API for the Developer to use Store’s features as well as a connector allowing Store to push information to the Developer.

- **Dashboard Adapter:** This module provides connectivity with the Security Dashboard component either in the form of an API for the Dashboard to use Store's features as well as a connector allowing Store to use Security Dashboard's functionalities.
- **Orchestrator Adapter**: This module provides connectivity with the Orchestrator component either in the form of an API for the Orchestrator to use Store's features as well as a connector allowing Store to use Orchestrator's functionalities.
- **Trust Monitor API**: This module provides connectivity to Trust Monitor component in the form of an API.
- **DARE API**: This module provides connectivity to DARE component in the form of an API.

3.2.2. Implementation details

To cope with the SHIELD security requirements, to foster VNF reuse and remove SHIELD applicability barriers, a package format tailored for SHIELD is defined. This package format, described in Annex G, extends existing VNF formats by introducing:

- a security manifest to ensure VNF tamper-proofing,
- a digitally-signed security manifest to prove provenance and integrity,
- support for including vNSFO-specific VNF package format,
- a `.tar.gz` package format to enclose everything.

REST API Services will be used to expose an interface to access the Store internal features. Further specifications comprising the envisioned APIs can be found in "Annex D Application Programming Interfaces (APIs)".

The REST backend is leveraged by Eve [44], a REST API framework which provides Flask [45] for RESTful support, Cerberus [46] for JSON validation, and MongoDB [47] for the actual vNSF & NS data store.

The services provided by the Store operate on a Role Based Access Control, resorting to a token-based authentication mechanism. To use an endpoint the caller must provide a token which the Store validates in the SHIELD authorization and authentication system. Only valid users with the required role are allowed through.

A Behaviour Driven Development [48] methodology is introduced for testing the Store. The option for this methodology ensures meeting the goals of (i) validating the Store behaviour from an API consumer's perspective, and (ii) serve as documentation for describing the features available and the intended operational scenarios.

3.2.3. Requirements mapping

Requirement	Requirement name	Requirement description
PF02	vNSF lifecycle management	The platform SHALL be able to manage the full lifecycle of vNSFs (on boarding, instantiation, chaining, configuration, monitoring and termination).

S_SPEC_01	The Store provides the Developer with an interface to onboard a vNSF and the Security Dashboard with another interface to onboard NSs. It also provides an interface to vNSFO to query vNSF and NS information during instantiation. The remaining states for the vNSF lifecycle management are outside the scope of the Store.	
PF10	vNSF validation	The store SHALL validate that the image of a vNSF is not manipulated, faked or invalid.
S_SPEC_02	At the time of a vNSF submission by the Developer the Integrity Checker ensures that the vNSF content is trusted and stores (amongst other data) the hash(es) for the vNSF image(s) which can be provided upon request for integrity checks by other components.	
PF11	vNSF attestation	The platform SHALL check the provenance and integrity of a vNSF and associated policies, before it starts to operate.
S_SPEC_03	When the Developer submits a vNSF the Integrity Checker validates the digital-signature associated with it to verify the provenance of the submitted data and analyses its integrity to ensure it wasn't tampered with in any way. This data is stored and can be provided upon request for attestation purposes to other components.	
PF15	Service store	The store SHALL allow selecting security services from the catalogue.
S_SPEC_04	A record of the successfully onboarded Network Services is kept by the Catalogue. These security services are provided upon request through the Store’s interfaces.	
PF17	Interoperability	The platform SHALL expose openly-defined APIs for information exchange with third parties.
S_SPEC_05	The Store provides the interoperability features through APIs and connectors. The vNSF onboarding is accomplished by the Developer’s API, the NS onboarding and store-related GUI interaction is done by the Dashboard API, the vNSF and NS data concerning orchestration is provided by the Orchestrator API and the attestation-related data is conveyed by the Trust Monitor API.	
PF22	Management communication security	The platform SHALL encrypt all the management communications.
S_SPEC_06	The communication with the Store APIs is done using HTTPS, where the messages are encrypted using the Transport Layer Security protocol.	
Resource Requirements	1 virtual processor, deployed in 40GB storage/4GB memory as minimum	

3.3. Orchestrator

When analysing the Platform Requirements (described in D2.1), four well-known NFV MANO solutions were identified and analysed. These are OSM [3], TeNOR [49], SONATA [2] and OpenBaton [50]. OSM stems from industrial community, whilst TeNOR, SONATA and OpenBaton have grown in the R&D environment.

To carry out the analysis, the accordance with the Platform Requirements (as defined in D2.1 [8] and D2.2 [9]) was examined, along with several extra indicators; from more subjective, like the extensibility and complexity degree in terms of development, to others such as its ongoing and future roadmap as well as its community. When considering how appropriate are the provided features to the SHIELD's Platform Requirements, OSM and TeNOR provide mostly the same capabilities; with more support by the former to extra VIMs and SDN controllers, and on monitoring and operational capabilities on the latter. SONATA and OpenBaton show a focus on specific aspects (the former focusing on identity management, the latter in service operations). At the time of the analysis, the sampled orchestrators were each at different stages of development; with SONATA under development and OpenBaton a more consolidated orchestrator. Both provide advanced features on their field of focus and provide extensive documentation. A detailed analysis can be found in the "Annex E Technology Selection". After evaluating the indicators and prioritising the Platform Requirements, the community and available support, the consortium decided to use OSM as the base vNSFO for SHIELD.

3.3.1. Specifications

The low-level specifications of the subcomponents of the Orchestrator are provided below:

NFVO-specific

- NS and VNF Manager: The NS and VNF Managers support issuing the following operations on the NSs and vNSFs:
    - Instantiation: initial deployment of the vNSF instances (contained on the deployed NS), according to the initial configuration defined by the NFVO during deployment or provided by the vNSF or NS descriptors. The operations on the vNSFs are delegated to the VNF Manager
    - Scaling: increase or decrease of the NS capacity according to the auto-scaling policies defined per vNSF and NS in their descriptors. The scaling can result in increasing/decreasing capacity per vNSF, creating or terminating vNSF instances and adjusting the number of links between vNSFs
    - Termination: release any given NS instance and its associated resources (vNSF instances, NFVI-related resources, connecting links between vNSFs)
- Configuration VNF Manager: This subcomponent performs part of the duties of the Entity Manager (EM); as it transfers the policies received via the Dashboard interface (in the vNSFO) to a specific instance of a vNSF running in the VIM. The vNSFO communicates with the vNSF to be configured by means of the management network, visible to both vNSFO and vNSFs.
- NFVI Connectors: the connectors and/or plug-ins available in OSM allow communication with different VIMs (OpenStack, OpenVIM, VMWare vCloud Director, Amazon Web

Services) and SDN controllers (OpenDayLight, ONOS, FloodLight). Prior configuration and persistence of some initial data, configuration or credentials must be performed.

- Registries: Data is persisted for operational purposes. The NS and vNSF Managers persist the NS and vNSF records (corresponding to specific deployed instances), whereas the Configuration VNF Manager keeps a registry of the actions and data (here, policies) sent to the vNSFs. The NFVI connectors also require saving some information to be able to connect and manage the VIM and managers in the NFVI (like the SDN controller).

SHIELD-specific

- Interfaces:
    - Store API: This exposes a write-only API so that the operations for onboarding or removing a specific package can be issues towards the NFVO instance, available as part of the vNSFO.
    - Dashboard API: This provides a write-only API that allows to configure a specific type of NS. The internal logic determines the feasibility of direct configuration of a deployed vNSF instance as part of such service, or the full deployment and consequent configuration process.
    - Dashboard Connector: This exposes a read-only API that provides necessary information on the resources in the NFVI. This can be used by the Security Dashboard to complement its knowledge on the topology.
    - Trust Monitor Connector: This attacks a write-only API that expects data on the NFVI and running instances.
    - Trust Monitor API: This exposes a read-only interface for the Trust Monitor to inform on the network, flow tables and list of active physical and virtual nodes. This input is used by the TM to complement its information on the NFVI, regarding the attestation procedure.
    - DARE API: This exposes a read-only interface for the DARE so as to obtain information on the physical nodes, the running and available vNSFs, etc. This interface is partly similar to that interface for the Trust Monitor.
- Logic
    - Onboard: This process starts once the Store receives a SHIELD package, validates it and request onboarding of the contained NFVO-specific package. The vNSFO then interacts with the NFVO to perform the onboarding, which registers the package into the vNSFO. The vNSFO allows interaction with the VIM(s) in order to register the image of the vNSFs.
    - Infrastructure: These interactions have the purpose of, amongst others, covering the definition of the physical infrastructure and virtual nodes that belong to the NFVI. This is used to share data of the NFVI to other interested components, such as the Trust Monitor.
    - Configuration: Using the management network defined between the VIM and the NFVO, the orchestrator is able to proxy to the vNSFs the specific middle-level policies that are conveyed via the Dashboard and the vNSFO, sequentially. Such policies are inserted into a given vNSF and converted internally to a low-level configuration action.

3.3.2. Implementation details

The vNSFO is based on the OSM solution. The following software modules and technologies will be used to fulfil the orchestration:

- Service Orchestrator (SO):
  - Acting as the NS Manager, RIFT.ware provides end-to-end network service orchestration, abstracting from computing resources, and provisioning lifecycle management and interconnection of VLs.
- Resource Orchestrator (RO):
  - OpenMano enables operations related to the allocation of resources from the NFVI and for the configuration through the Configuration VNF Manager. It provisions resources as needed, potentially interacting with multiple VIMs and SDN controllers. Along with SO, these conform the NFVO entity in the ETSI NFV architecture.
- vNSF Configuration and Abstraction (VCA):
  - Generic vNSF Manager allowing the initial vNSF configuration. It relies on Canonical's Juju charms and cloud-init to provide instructions to the vNSFs either before or during deployment. This has a partial correspondence to part of the Entity Manager in the ETSI NFV architecture.

These modules can be mapped to the ETSI NFV architecture as depicted in Figure 24:

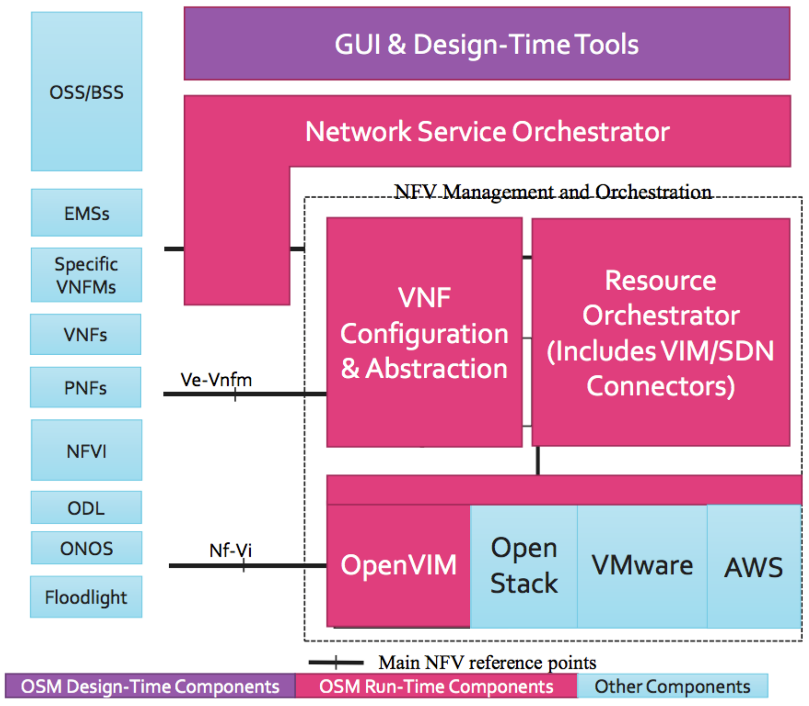

GUI & Design-Time Tools
Network Service Orchestrator
OSS/BSS
EMSs
Specific VNFMs
VNFs
PNFs
NFVI
ODL
ONOS
Floodlight
NFV Management and Orchestration
VNF Configuration & Abstraction
Resource Orchestrator (Includes VIM/SDN Connectors)
Ve-Vnfm
Nf-Vi
OpenVIM
Open Stack
VMware
AWS
Main NFV reference points
OSM Design-Time Components
OSM Run-Time Components
Other Components

Figure 24: OSM mapped to ETSI NFV architecture.

More details on the implementation and deployment details can be found in the documentation and whitepaper for Release TWO that SHIELD deploys [51] [52].

The details on the specific development for the interfaces and connectors described earlier are provided in the "Annex D Application Programming Interfaces (APIs)".

3.3.3. Requirements mapping

Requirement	Requirement name	Requirement description
PF01	vNSF and Network Service (NS) deployment	The platform SHALL be able to deploy the vNSFs in different PoPs and domains. The deployment can occur within internal or external premises.
O_SPEC_01	The NS and VNF Managers can initiate the deployment of the vNSFs in the different PoPs; considering these are previously registered into the NFVO.	
PF02	vNSF lifecycle management	The platform SHALL be able to manage the full lifecycle of vNSFs (on boarding, instantiation, chaining, configuration, monitoring and termination).
O_SPEC_02	The NS and VNF Managers can control the different stages in the lifecycle of the NSs and vNSFs.	
PF03	vNSF status management	The operator SHALL be able to control the lifecycle via a graphical user interface. The vNSF lifecycle should support events like DEPLOY, START, STOP, MODIFY, DELETE.
O_SPEC_03	The NS and VNF Managers may receive lifecycle events to deploy or instantiate, run, stop or delete the vNSFs. Configuration may be performed via the Configuration VNF Manager.	
PF07	Service elasticity	The platform COULD provide the mechanism to allow scalability of the vNSFs.
O_SPEC_04	The NS and VNF Managers provide the capability to request a specific NS or vNSF to adapt (scale) to its operational conditions.	
PF11	vNSF attestation	The platform SHALL check the provenance and integrity of a vNSF and associated policies, before it starts to operate.
O_SPEC_05	The vNSFO receives the associated policies for a given vNSF and is able to apply them; as well as providing attestation-related data to the Trust Monitor.	
PF13	Mitigation	The platform SHALL be able to trigger, in the case of an event, proper actions to mitigate the threat.
O_SPEC_06	As the result of an accepted suggestion by a user in the Dashboard, the NS and VNF Managers receive and distribute requests to deploy mitigation NSs.	
PF22	Management communications security	The platform SHALL encrypt all the management communications.
O_SPEC_07	The vNSFO implements encrypted connections via TLS for all its interactions with other components of the platform.	

NF03	Scalability	The platform SHALL be expandable by adding nodes in the network infrastructure, to increase capacity.
O_SPEC_08	The vNSFO is aware of new nodes registered into the NFVI, as required for attestation purposes by the Trust Monitor.	
Resource Requirements	The vNSFO requires at least 4 virtual processors, 60 GB of storage and 8 GB of RAM and a single interface with Internet access. Recommended specifications include 8 virtual processors, 100 GB of storage and 16 GB of RAM; as well as the single interface with Internet access.	

3.4. Trust monitor

The general architecture and design of the Trust Monitor has been defined according to the Platform Requirements, as defined in D2.1. This section aims to describe the specifications of the low-level functionalities that will be developed within the Trust Monitor sub-components.

3.4.1. Specifications

The low-level specifications of each subcomponent of the Trust Monitor are reported as follows, as well as the mapping of the specifications to the PFRs. The specifications may be subject to minor modifications during the development stage.

Verifier: The Verifier is the central sub-component of the Trust Monitor. It manages different functionalities:

- Registration of a node
- On-demand attestation of a node
- Periodic attestation of the nodes in the NFVI
- Notification of attestation failure to both the DARE and the vNSFO

The **registration phase** is needed to setup the attestation process with each NFVI PoP composing the network infrastructure. Each node of the NFVI should be properly configured to enable its interaction with the TPM and to start measuring the software running into it. The remote attestation procedure is performed during the initial attestation of newcomers and the periodic attestation tasks. It requires the Verifier to perform the following operations:

1. Send an attestation request to the node, including a nonce for freshness of the response
2. Validate the response
3. Extract the software measurements from the integrity report, consisting of the software and configuration utilised by both the host and the vNSFs running into it
4. Verify the integrity measurements of the host against the reference values contained in the Whitelist Database
5. For each vNSF, verify the integrity measurements against the known digests contained in the vNSF security manifest
6. For SDN-controlled switches, verify the SDN forwarding rules with regards to the expected one (in the SDN controller)

The Verifier can assess the measurements of the host by leveraging the Whitelist Database functionality. The known measurements of each vNSF can be retrieved via the API exposed by the vNSF Store. In case of failure during attestation, the Verifier sends a notification by leveraging the APIs provided by both the vNSFO and the DARE components. Periodic attestation should be performed by an internal task that uses the API offered by the vNSFO to retrieve the "map" of the status of running nodes in the NFVI.

DARE Connector: The Trust Monitor should be able to collect relevant information from the NFVI in real time to verify the nodes' software integrity. This information is used to detect security incidents regarding misuse of a node. In case of failure upon attestation, a security event is sent by the Trust Monitor to the DARE. This information is logged by the DARE and it could also be shared with a third entity.

Whitelist Database: The database contains the complete data of the executables allowed on the attested platforms. More specifically, for example on Linux-based platforms, it contains the digest, the full path name, and the executable's packages (grouped by distributions and architecture). Given the supported distributions and architectures, the database is initialised and updated periodically by downloading the packages' lists from their official repositories. Alternatively, the database can be updated with release information for components that do not come from public repositories. Additionally, the database should store the history of each package, reporting the information about its updates (e.g. the type of update). Given the packages' history, the Verifier verifies the IMA log at different trust levels:

- **Level 1:** TPM and IMA measurement in the node is running correctly
- **Level 2:** In addition to Level 1, all the software is found in the reference database but there is at least one with a known security vulnerability
- **Level 3:** In addition to Level 2, at least one binary has a known functional bug
- **Level 4:** In addition to Level 3, no known security vulnerabilities or functional bugs are found in the measured software

vNSF Store Connector: This subcomponent allows the retrieval of the security manifest for each vNSF to be attested.

vNSFO Connector: The vNSFO is in charge of terminating nodes of the NFVI if their execution cannot be trusted. Therefore, the Trust Monitor is in charge of notifying both the vNSFO and the DARE in case of remote attestation failure. In addition, the Trust Monitor should have a clear view of the vNSFs and NFVI PoPs deployed in the SHIELD infrastructure, in order to perform the periodic attestation of running nodes. To do so, it will leverage a specific functionality offered by the vNSFO API.

Newcomer Attestation API: The sub-component exposes an API for on-demand registration and attestation of newcomers in the NFVI.

Management API: The sub-component exposes a read-only API for checking the status of the Trust Monitor and retrieving relevant information about the attestation of the infrastructure.

3.4.2. Implementation details

The Trust Monitor implementation starts from components that have been developed in the EC-funded project SECURED [4]. More specifically, the following technologies could be reused:

- Third-party Verifier based on Open Attestation v1.7 [5],
- Whitelist Database based on Apache Cassandra 2 [6],
- SDN-enabled switch attestation prototype [7],

These technologies, representing the starting point for the development stage, are bound to a Linux CentOS 7 environment equipped with TPM 1.2 device. The development efforts in the project are aimed to enrich the already available software with the SHIELD-specific APIs.

Additionally, the Trust Monitor should be able to support TPM 2.0-enabled hardware, meaning that the attestation framework needs further improvements. Regarding this point, the OpenCIT [53] framework, developed by Intel, will be exploited as an evolution to the Open Attestation framework.

As stated in the official website of the project, Open Attestation no longer receives any update and it does not provide support for the TPM 2.0 devices. On the opposite side, OpenCIT does not support the integrity report workflow (as of March 2018), meaning that further improvements are needed over the mainstream version. The details on the specific development for the interfaces and connectors described earlier are provided in the "Annex D: Application Programming Interfaces (APIs)".

3.4.3. Requirements mapping

Requirement	Requirement name	Requirement description
PF08	Platform expandability	The platform SHALL be easily extended to support new security services.
T_SPEC_01	The Trust Monitor provides documented APIs and interfaces to enable the interaction with the different components. In addition, the component provides a generic client-service workflow to attest the nodes in the NFVI.	
PF11	vNSF attestation	The platform SHALL check the provenance and integrity of a vNSF and, when applicable, its associated policies, before it starts to operate.
T_SPEC_02	The Trust Monitor should attest the vNSFs deployed on top of a host in the NFVI and provides notifications to both DARE and vNSFO.	
PF16	History reports	The platform SHALL generate reports of past incidents based on historic data.
T_SPEC_03	The Trust Monitor contributes to the definition of reports of past incidents, as it will provide notifications to both the DARE and the vNSFO to enrich the logs of occurring incidents.	
PF19	Network infrastructure attestation	The platform SHALL verify that the network infrastructure executing the NSs is in a trusted state (network elements, server identity, software and their configuration).
T_SPEC_04	The Trust Monitor attests the software integrity of the network infrastructure and provides notifications to both DARE and vNSFO.	

PF22	Management communications security	The platform SHALL encrypt all the management communications.
T_SPEC_05	The Trust Monitor implements encrypted connections via TLS for all its interactions with other components of the platform.	
NF01	Response time	The platform SHALL report the incident within a relatively short time (in the order of seconds)
T_SPEC_06	The Trust Monitor periodically attests the nodes in the NFVI (in the order of seconds) to identify any occurring incidents and report them to both DARE and vNSFO. The bottleneck for minimising the latency between two subsequent attestations is the latency introduced by the usage of TPM, as it registers the measurements in the node.	
NF07	Compliance to standards	The platform SHALL conform to well-established standards, in particular with respect to data export (e.g. STIX) and input (e.g. NetFlow).
T_SPEC_07	The Trust Monitor adopts well-established standards by the Trusted Computing Group to describe the formats of integrity reports.	
NF08	Deployment and support simplicity	The platform SHALL be easily installed and maintained, without the need of specific expertise.
T_SPEC_08	The Trust Monitor can be instantiated as a set of containers that are run and interconnected automatically by the container runtime (Docker). Specific subcomponents of the Trust Monitor can be deployed independently on different machines (e.g. the Whitelist Database and the Verifier).	
ERC07	Notification obligation	In the case of a breach in a component that processes personal data, the platform SHALL produce a breach notification. Data rectification or erasure should be accompanied with a notification to the data subject unless it is difficult or involves disproportionate effort, as per article 19 of the GDPR.
T_SPEC_09	The Trust Monitor sends a notification if it detects a breach in a SHIELD NS.	
Resource Requirements	The Trust Monitor requires at least 2 virtual processors, 100 GB of storage and 4 GB of RAM. Recommended specifications include 4 virtual processors, 100 GB of storage and 8 GB of RAM.	

4. Regulatory Compliance Specifications

Ensuring the regulatory compliance of the vNSF ecosystem is a key activity that removes barriers towards their adoption. This section discusses the regulatory framework that applies across EU member states that is relevant to the operational aspects of the vNSF ecosystem. Based on this analysis, SHIELD extracts the regulations-based specifications for the key components and illustrates how to start implementing compliance mechanisms. The key focus of this work is on:

- Privacy and Data Protection,
- Obligations of the service provider to Law Enforcement/CERTs,
- Net Neutrality and fair traffic classification,
- Non-discrimination and protection of the individual's rights against behavioural profiling.

A face-to-face meeting with SHIELD's Ethics Advisor Prof. Haralambos Mouratidis[1] took place in Athens (March 16th, at Space Hellas SA premises). Prof. Mouratidis provided valuable feedback on GDPR and overall legal compliance and on the definition of specifications.

4.1. EU regulatory framework

4.1.1. Analysis of EU regulatory landscape

General Data Protection Regulation: Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation) (Text with EEA relevance) [54]

The EU General Data Protection Regulation is in place to safeguard the rights of the data subjects and enable them to better control their personal data. The Regulation aims to alleviate the fragmentation in data protection law across EU member states and replace the previous Directive with a unified set of rules. The GDPR features an improved territorial scope since it applies to controllers/processors of personal data that are established in the Union, regardless of the location of the processing. **Article 4** makes the following definitions:

Key definitions in Article 4 with relation to SHIELD
1. **'personal data'** means any information relating to an identified or identifiable natural person ('data subject'); an identifiable natural person is one who can be identified, directly or indirectly, in particular by reference to an identifier such as a name, an identification number, location data, an online identifier or to one or more factors specific to the physical, physiological, genetic, mental, economic, cultural or social identity of that natural person; 2. **'processing'** means any operation or set of operations which is performed on personal data or on sets of personal data, whether or not by automated means, such as collection, recording,

[1] Prof. Haralambos Mouratidis, Director Centre for Secure, Intelligent and Usable Systems (CSIUS), Professor of Software Systems Engineering, School of Computing, Engineering and Mathematics, University of Brighton.

organisation, structuring, storage, adaptation or alteration, retrieval, consultation, use, disclosure by transmission, dissemination or otherwise making available, alignment or combination, restriction, erasure or destruction;

[…]

4. **'profiling'** means any form of automated processing of personal data consisting of the use of personal data to evaluate certain personal aspects relating to a natural person, in particular to analyse or predict aspects concerning that natural person's performance at work, economic situation, health, personal preferences, interests, reliability, behaviour, location or movements;

[…]

5. **'controller'** means the natural or legal person, public authority, agency or other body which, alone or jointly with others, determines the purposes and means of the processing of personal data; where the purposes and means of such processing are determined by Union or Member State law, the controller or the specific criteria for its nomination may be provided for by Union or Member State law;
6. **'processor'** means a natural or legal person, public authority, agency or other body which processes personal data on behalf of the controller;

[…]

11. **'consent'** of the data subject means any freely given, specific, informed and unambiguous indication of the data subject's wishes by which he or she, by a statement or by a clear affirmative action, signifies agreement to the processing of personal data relating to him or her;

The network data that are being processed by SHIELD components such as the vNSFs may include personal data in the form of IP addresses, emails, login credentials etc. SHIELD, however, does not profile a natural person's behaviour. SHIELD vNSFs do not inspect the contents of communications or assess personal aspects of a natural person's behaviour (e.g. buying patterns, religious beliefs, health etc.). The basic principles that underline the GDPR (Article 5) regard:

- The lawfulness, transparency and fairness of processing;
- The limitation of its purpose (data must be collected for clear and explicit reasons);
- The principle of data minimization (data collected should be adequate to perform the specific purpose but limited to what is necessary);
- The accuracy of the data;
- The minimization of storage that permits identification of the data subject for no longer than necessary;
- The security and confidentiality of the data.

Article 6 further analyses the lawfulness of processing while **Article 7** details the consent processes that should apply.

Article 6 Lawfulness of processing
1. *Processing shall be lawful only if and to the extent that at least one of the following applies:* (a) *the data subject **has given consent** to the processing of his or her personal data for one or more specific purposes;* (b) *processing is necessary **for the performance of a contract** to which the data subject is party or in order to take steps at the request of the data subject prior to entering into a contract;* (c) *processing is necessary for **compliance with a legal obligation** to which the controller is subject;*

(d) processing is necessary in order to ***protect the vital interests of the data subject*** *or of another natural person;*
(e) processing is necessary for ***the performance of a task carried out in the public interest*** *or in the exercise of official authority vested in the controller;*
(f) processing is necessary for the purposes of ***the legitimate interests pursued by the controller or by a third party, except where such interests are overridden by the interests or fundamental rights and freedoms of the data subject*** *which require protection of personal data, in particular where the data subject is a child.*

In the context of SHIELD, processing for the explicit purpose of security can considered lawful. Cybersecurity and protection of network infrastructures against intrusions and breaches can be considered as vital interests of the data subject and as tasks carried out in the public interest. The ISP that uses SHIELD vNSFs to secure their network need to inform their clients accordingly and ask for their consent within their contract. In the case of ISP offering services to organisations as SecaaS clients, the processing of network data is required to fulfil a contract. The SecaaS client needs to ensure that the personnel are informed of the specific network monitoring activities. **Article 7** of the GDPR also states that when consent is given in the context of a written declaration which also concerns other matters (e.g. a contract) it must be presented in a form that is easily distinguishable and comprehensible, otherwise the declaration will not be considered binding.

GDPR dedicates **Articles 12-23** to the description of the Rights of the Data Subject and how they shall be exercised, including the right for access, erasure, restriction, rectification, portability and the right to be forgotten. It further specifies, that if the stored information is not identifiable, then the data subject is responsible to provide additional information to identify their data. Portability is another important aspect, since it is **aligned with EU's competition law.** If a person's data are not portable among different instances of SHIELD, then customer lock-in conditions are created. The data subject rights however, do not apply if a component does not retain any data. **Articles 24-43** relate to the responsibilities of the data controller, the data processor, and establishes the role of the Data Protection Officer. This information should be transparent to the user as well. The data subject should be able to contact the DPO or the Data Processor regarding their data. **Article 26** describes the case for **Joint Controllers**. This case could be applicable to the SHIELD SecaaS use case (use case 2). The text also includes rules on data sharing **(Articles 44-50).** If data are shared with third parties (or monetized) the data subject should consent. **Cross-border data sharing** is not foreseen within SHIELD. **Sharing data with Law Enforcement or CERTs** should be enabled for alignment with other Directives as well. **Other issues covered in the GDPR include** the role of independent supervisory authorities **(Articles 51-59),** Liabilities and penalties **(Articles 77-84)** etc.

<u>Open Internet Regulation:</u> Regulation (EU) 2015/2120 of the European Parliament and of the Council of 25 November 2015 laying down measures concerning open internet access and amending Directive 2002/22/EC on universal service and users' rights relating to electronic communications networks and services and Regulation (EU) No 531/2012 on roaming on public mobile communications networks within the Union (Text with EEA relevance). [55]

The Open Internet Regulation establishes the circumstances where traffic classification and management are legitimate. It lays down specific net neutrality rules and governs the way ISPs may choose to manage the traffic that passes through their networks, while ensuring equal and non-discriminatory treatment of traffic. Specifically, the following aspects are particularly relevant to SHIELD:

(8) When providing internet access services, providers of those services should treat all traffic equally, without discrimination, restriction or interference, independently of its sender or receiver, content, application or service, or terminal equipment. According to general principles of Union law and settled case-law, comparable situations should not be treated differently, and different situations should not be treated in the same way unless such treatment is objectively justified.
(10) Reasonable traffic management does not require techniques which monitor the specific content of data traffic transmitted via the internet access service.
(12) Traffic management measures that go beyond such reasonable traffic management measures may only be applied as necessary and for as long as necessary to comply with the three justified exceptions laid down in this Regulation.
(13) First, situations may arise in which providers of internet access services are subject to Union legislative acts, or national legislation that complies with Union law (for example, related to the lawfulness of content, applications or services, or to public safety), including criminal law, requiring, for example, blocking of specific content, applications or services.
(14) Second, traffic management measures going beyond such reasonable traffic management measures might be necessary to protect the integrity and security of the network, for example by preventing cyber-attacks that occur through the spread of malicious software or identity theft of end-users that occurs as a result of spyware.
(15) Third, measures going beyond such reasonable traffic management measures might also be necessary to prevent impending network congestion, that is, situations where congestion is about to materialise, and to mitigate the effects of network congestion, where such congestion occurs only temporarily or in exceptional circumstances.

Article 3 further states that traffic management must be reasonable, transparent, non-discriminatory and proportionate. **Article 4** details how providers of internet access services shall be transparent in their contracts about traffic management; hence, traffic management through SHIELD for cybersecurity purposes should be included. **Article 5** also mentions that national authorities should be able to monitor compliance with this Directive and record their findings.

The application of traffic classification and rate limiting within the context of SHIELD is lawful since it does not restrict users' access based on arbitrary or business-oriented decisions. In SHIELD, cybersecurity is the sole purpose of traffic classification and any rate-limiting measures are attached to a security event which is parsed by the recommendation engine and reported in the dashboard. This allows for a level of transparency when applying such measures to limit, redirect or block specific types of traffic and enables monitoring from national authorities. Furthermore, the application of rules can be restricted to suspicious IP addresses or applications/protocols and can thus be considered proportionate since the decision to apply them includes only offending flows that were detected in the DARE. Rate limiting can and should be rolled back when there is no additional danger to the security of the network. Furthermore, the SHIELD vNSFs do not investigate the communication contents for traffic management.

ePrivacy Directive: Directive 2002/58/EC of the European Parliament and of the Council of 12 July 2002 concerning the processing of personal data and the protection of privacy in the electronic communications sector (Directive on privacy and electronic communications) [56]

The ePrivacy Directive, also known as the EU Cookie Law, sets the rules for the collection of cookies and ensures confidentiality of electronic communications. At the moment, there is a proposal [57] for a revision of the ePrivacy Directive, to better align it with the GDPR, take into account continuing technical innovation, and to transform it into a Regulation. This would mean that the EU Member States would implement the Regulation as-is, as opposed to a Directive which can be implemented in any way considered suitable by the Member States. The proposal for the Regulation was released on January 2017. Regarding the applicability of the ePrivacy Directive and the future ePrivacy regulation, SHIELD makes the following assumptions:

- The SHIELD vNSFs are not applications that require cookies to provide a user experience, they do not track user preferences through cookies and hence do not require SHIELD to ask the user for cookie consent.
- Cookies, however, can be part of the network traffic and as they provide identifiability, SHIELD considers the GDPR stipulations for their protection.
- Profiling of a user's behaviour through cookies is not considered in any SHIELD vNSF. Otherwise, consent and additional safeguards to ensure the data subject's rights and non-discrimination should be in place. protection of communication contents under the new regulation will apply to telco traffic (e.g. SMS), as well as other digital communications providers (e.g Skype, WhatsApp etc.).
- The confidentiality of personal communications is also relevant to SHIELD, as the vNSFs do not compromise encrypted safe communications.

Regarding the proposal for the ePrivacy Regulation, a new component is the protection not only of communications content but also of **communication metadata**. SHIELD already applies GDPR protections to cookies and information metadata. Pending finalisation of the ePrivacy Regulation, additional protections can be considered.

<u>Data protection in criminal investigations:</u> Directive (EU) 2016/680 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data by competent authorities for the purposes of the prevention, investigation, detection or prosecution of criminal offences or the execution of criminal penalties, and on the free movement of such data, and repealing Council Framework Decision 2008/977/JHA. [58]

<u>Network Information Security Directive:</u> Directive (EU) 2016/1148 of the European Parliament and of the Council of 6 July 2016 concerning measures for a high common level of security of network and information systems across the Union [59]

Although these directives do not apply directly to SHIELD, they are relevant as service providers may be required to cooperate with law enforcement in a criminal investigation or with appropriate cybersecurity agencies in case of a cyberattack. A vNSFs may expose APIs for exchange of information with relevant third parties under these directives, although the exact use of such APIs would be defined by the ISP or SecaaS client and their internal policies regarding statutory process (unless access is court-mandated). Specifically, the NIS Directive aims to develop the principles for European cyber-crisis cooperation. Since it is a Directive, the Member States can select the specific of its implementation, leading to concerns on fragmentation and disparities among Member States. NIS states that a certain level of cooperation and cyber security readiness is expected from operators of critical services, defined as (article 4):

(a) an entity provides a service which is essential for the maintenance of critical societal and/or economic activities;

(b) the provision of that service depends on network and information systems; and

(c) an incident would have significant disruptive effects on the provision of that service.

This applies to critical infrastructures (including banking, health, transport etc.), providers of telecommunication basic services (DNS providers etc.), digital service providers (e.g. marketplaces), cloud infrastructure providers etc. Therefore, it is applicable to the SHIELD concept in the sense that SHIELD should improve communication of cyber incidents among operators and cyber security response teams and agencies.

Non-discrimination:

Council Directive 2000/78/EC of 27 November 2000 establishing a general framework for equal treatment in employment and occupation. [60]

European Charter of Fundamental Human Rights [61], esp. Article 8(1) on the protection of personal data

Treaty of Amsterdam [62] (1997/1999 establishing the protected grounds against discrimination) & Treaty of Lisbon [63] (2007/2009 making the ECHR Bill of Rights legally binding)

Council of Europe recommendations on profiling: Recommendation CM/Rec(2010)13 of the Committee of Ministers to member states on the protection of individuals with regard to automatic processing of personal data in the context of profiling. [64]

Although the non-discrimination body of law in the EU regards access to employment, education etc. which are out of the scope of SHIELD, we can consider some basic principles and definitions to be free-standing. Access to the Internet can be regarded as a basic service that should be available to all citizens and any discriminatory practices should be abolished. Although SHIELD vNSFs do not profile the user's behaviour for cybersecurity, some definitions should be in place, for future reference:

- The entry into force of the Treaty of Amsterdam in 1997, enabled the European Commission to legislate on non-discrimination based on **defined protected grounds** which **include gender, age, race, ethnicity, religion, belief, age, disability and sexual orientation**. The GDPR considers data that may expose these aspects of the data subject as "special category" data.
- Protection against discrimination is not only present in EU Law but also within the **European Charter of Human Rights (ECHR)** that was proclaimed by the European Union and the Member States in 2000. The ECHR declared the fundamental human rights to be protected and became legally binding after the 2009 Treaty of Lisbon.
- Most definitions in EU law and ECHR regard cases of direct discrimination. The EU Agency of Fundamental Rights (FRA) [65], however, further defines **indirect discrimination**, when a rule that appears to be neutral affects a specific group of citizens in a significantly more negative way, by comparison to others in a similar situation. It also defines **harassment and instruction to discriminate** as violating the dignity of a person.

Hence, any data processing component that profiles aspects of the data subject with respect to these protected grounds, should have safeguards in place to ensure that processing is lawful and that such information cannot be misused and lead to discriminatory practices. The Council of Europe has published a recommendation on safeguards for processing that leads to profiling, although this predates the GDPR and there was no legal definition of profiling at the time.

4.1.2. Basic assumptions on the stakeholder roles and obligations

In order to properly map what the roles and obligations of each stakeholder, it is first necessary to understand the ecosystem of actors across SHIELD use cases. The main stakeholders that can be immediately identified include:

- The service provider that deploys the SHIELD vNSFs (e.g. an ISP or SecaaS provider),
- The vNSF developer that offers their products through a marketplace,
- Natural persons that purchase internet or SecaaS services from a provider (e.g. a person who uses the internet in their home environment)
- Organisations (legal persons) that purchase internet or SecaaS services from a provider (e.g. a company that purchases DDoS protection for their network) and the natural persons that use their network,
- The Data Protection Authorities, Law Enforcement, CERTs etc. and all organisations that might interface with the Data Controllers in case of a cyber security incident, data breach etc.

The first step is to identify which actors take up the role of the Data Controller, the Data Processor, the Data Protection Officer and the data subject, as well as their specific obligations.

Obligations of the Data Controller: The GDPR considers data subjects that are natural persons. Across all use cases that role is assumed by the users that connect to a network protected by SHIELD vNSFs. When a service provider utilises SHIELD vNSFs, it is their obligation to obtain **consent** from their clients (i.e. the data subjects). Consent can be given in the form of the contract between e.g. an ISP and the client purchasing services for a home network. Article 7 of the GDPR states that in this case all information "must be presented in a form that is easily distinguishable and comprehensible, otherwise the declaration will not be considered binding." The service provider thus takes up the role of the **Data Controller** and appoints the **Data Processor** and **Data Protection Officer,** whose contact information should be accessible to the data subjects. This is applicable in **Use Case 1**, when the ISP uses the vNSFs to secure their own network infrastructure. Use Case 2 (SecaaS), however, assumes a different ecosystem. When a private or public organisation purchases Security-as-a-Service from a provider, **both the client and the provider take up the role of the Data Controller**. The data subjects are the natural persons using the client organisation's network (e.g. employees etc.). This use case falls under the case of **Joint Controllers.** In such a case, the existence of joint controllers and their association should be transparent, and the data subjects should have access to the related data protection information. **Use Case 3** assumes that the Data Controller should provide information to Law Enforcement agencies, CERT/CSIRT teams etc. in case of a major cybersecurity incident. It is the responsibility of the vNSF provider to include such interfaces in the vNSF if necessary, although its use hinges on the operational procedures adopted by the Data Controller. The way that statutory processes are being implemented by the Data Controller, should overlap with the vNSF, hence the Data Controller should remain in control of the reporting. The Data Controller is also obligated to refer to the Data Protection Authority to obtain consultation, authorisations or to report a data breach.

Obligations of the vNSF developer: The vNSF developer that offers their products through a Store or marketplace is obligated to provide the full specifications regarding how data processing is performed within the vNSF, as well as all the APIs and interfaces that allow data sharing. Using data minimisation practices within the vNSF, data encryption or anonymisation/pseudonymisation is the obligation of the developer. It is the obligation of the service provider that purchases and instantiates the vNSF to provide information on how it is used (i.e. which data sharing APIs are being used, etc.) to their clients. If data outputs from the vNSF are being used for further processing by an ISP, it is the obligation of the ISP to provide the specifications for the additional processing. The vNSF developer is required to analyse which types of personal information can be viewed by the vNSF (e.g. packet headers, email accounts, device IDs etc.) and specify if identifiability of the data subject can arise from the processing within the vNSF. In essence, the vNSF developer should complete a Data Protection Impact Assessment for each vNSF product. The following subsection (Subsection 4.2) further includes some best practices that could be adopted by the vNSF developer (and SHIELD overall).

Obligations of the Data Processor and Data Protection Officer: The system administrator that onboards, instantiates and manages the vNSF acts as the **Data Processor**. The Data Processor should be able to ensure that the rights of the data subject are being respected. Hence, the Data Processor should have available interfaces to erase, rectify etc. personal data when asked, either through the Dashboard or directly through the management interfaces of the vNSF. If the vNSF device does not retain data, the existence of such an interface might be irrelevant. If the data retained within the vNSF are not identifiable, the GDPR states that the data subject must provide a way to identify their data. The **Data Protection Officer** should serve as a contact point between the Data Controller and the appropriate **Data Protection Authority**. In case of a data breach, both the data subjects and the Data Protection Authority should be notified.

ePrivacy Compliance: In the context of ePrivacy, the SHIELD vNSFs do not utilise cookies to offer a user experience hence there is no obligation from the vNSF developer or ISP to provide cookie disclaimers. The network traffic, however, that passes through the vNSF might contain cookie information. In such a case, cookies receive the same level of protection as any identifiable personal data under the GDPR. SHIELD can consider additional measures to protect cookies and other communication metadata, should the ePrivacy regulation require additional protections for cookies and other communication metadata.

Net Neutrality: Since the Dashboard logs remediation actions and the associated security events, there is a certain level of transparency to remediation measures that apply rate limiting or blocking actions to a specified type of traffic. SHIELD and especially the vNSF developers, are obligated to ensure that appropriate interfaces are available to roll back these remediation measures once the security incident has ended. This can be performed either through the Dashboard or through the vNSF management interface.

Non-discrimination: The SHIELD vNSFs do not profile the user based on their network traffic or look into personal communication contents (e.g. messages, emails etc.). Hence, the risk of discrimination on the grounds of the Amsterdam Treaty is minimal. If additional processing takes place outside of the vNSF (either in DARE or another third-party component) it is the obligation of the ISP or SecaaS client to specify whether the user is being profiled and to ensure that the profiling results are not being misused. There needs to be assurance that the profiling information is not being used to deny access to basic services to a natural person based on their gender, ethnicity, religious, political views etc. This extends to the provision of internet access, to employment practices such as termination of a contract etc. In case of a remediation action that denies access to a user due to a security event, there is a level of transparency since the dashboard logs the security events attached to a remediation action. SHIELD and the vNSF developer, however, are obligated to ensure that in case of a false positive detection of an attack (i.e. when a user has been denied access to a network due to a false identification of a cybersecurity incident), the remediation action can be rolled back and access to a user can be restored. In SHIELD this can be implemented via either the Dashboard or directly through the management interface of the vNSF.

4.2. Best practices

Value-sensitive design (VSD) is built around the simple concept of designing technology to encompass human values and principles [66]. VSD addresses common design issues by taking into account the ethical values of the involved stakeholders. **Privacy by design** is an example of VSD that focuses on privacy during the entire development of a product. The way that Privacy-by-design is achieved depends on the application and technologies involved although the design is guided by a set of foundational principles [67]:

1. **Proactive not reactive; preventative not remedial:** the approach should be characterised by proactive measures that come before-the-fact.
2. **Privacy as the default setting:** Even if a user does not set specific policies, their privacy is still, automatically protected.
3. **Privacy embedded into design:** Privacy is integral to the system without diminishing functionality and not "bolted on as an add-on".
4. **Full functionality – positive-sum, not zero-sum:** False dichotomies ("privacy vs security") should be avoided and no unnecessary trade-offs should be made.
5. **End-to-end security – full lifecycle protection:** Strong security measures are essential and should apply to the entire data lifecycle. This extends to the introduction of Security-by-design.
6. **Visibility and transparency – keep it open:** Trust is easier to build when there is transparency and the stated promises can be verified across all stakeholders.
7. **Respect for user privacy – keep it user-centric:** Keep in mind the interests of the individual and provide privacy defaults, notices and empowering user-friendly options.

The GDPR further includes a definition of **data protection by design and by default**:

"In order to be able to demonstrate compliance with this Regulation, the controller should adopt internal policies and implement measures which meet in particular the principles of data protection by design and data protection by default."

It dedicates Article 25 on data protection principles such as data minimisation, i.e. ensuring that only the data that are required for processing are used.

A **Data Protection Impact Assessment** [68] is the process where the data protection risks of a project are evaluated. The DPIA should include information on the nature and characteristics of the data processing, the necessity and proportionality of processing, an assessment of risks and the mitigation measures that might be applicable. The Data Protection Officer of an organisation should review the DPIA with the assistance of the Data Processor and notify the National Data Protection Authority if further authorisation or consultation is required. Article 35 of the GDPR provides the minimum requirements for a DPIA. It needs to provide:

- a systematic description of the planned processing operations and the purposes of the processing, including, where applicable, the legitimate interest pursued by the controller,
- an assessment of the necessity and proportionality of the processing operations in relation to the purposes,
- an assessment of the risks to the rights and freedoms of data subjects,
- the measures intended to address the risks, including safeguards, security measures and mechanisms to ensure the protection of personal data and to demonstrate compliance with the GDPR with regard to the rights of data subjects and other persons concerned.

Hence, the vNSF developers need to perform a DPIA and assess the data privacy risks of each vNSF. To that extent, SHIELD reviews possible privacy risks, and lists specific mitigation measures. This work, along with the analysis in 4.1.1, sets the basis for a set of compliance specifications that are generated specifically for vNSF products, since no available DPIA template covers the specificities of SDN/NFV technology. The main aim of this work is to ensure that DPIA practices are adopted by the SHIELD partners and offer support for the development of GDPR compliant future products. Therefore, D3.2 presents an overview of potential risks and mitigation measures and provides detailed specifications regarding the amount of data processing activities of the vNSF ecosystem. Similar work is also included in D4.2 regarding the DARE components.

The **Open Web Application Security Project** (OWASP) maintains a list of top 10 Privacy risks and related countermeasures [69]. In the following table an analysis of these risks is presented, in the context of the SHIELD vNSF ecosystem, and an adaptation of the countermeasures is provided so as to account for the platform's specificities.

Table 2 Privacy risks and countermeasures.

Privacy Risk	Application to SHIELD	Countermeasures
P1 Web Application Vulnerabilities	Although the vNSFs are not web applications, we can consider that failure to suitably design it or apply a fix to known security vulnerability can result to a privacy breach, especially if the vNSF holds any personal data stored. Attention should be payed to the virtual machine's hypervisor technology as well. This also applies to the Store as it should be hardened against known vulnerabilities.	Perform penetration tests, monitor vulnerabilities (including those related to the vNSF hypervisors), train developers in secure development, install updates, fixes etc.

P2 Operator-sided Data Leakage	Failure to prevent a data leak can result in loss of confidentiality. This applies especially to any vNSF that stores information, as well as the SHIELD Store. The Trust Monitor can identify untrusted vNSF and NFVI and provide early warning.	Access control and Identity management following the principle of least privilege, strong encryption for personal data, awareness training, data classification and handling policies, data leak prevention/early warning, privacy-by-design, data anonymization/pseudonymization.
P3 Insufficient Data Breach Response	The persons affected by a data leak should be informed. Immediate action should be taken to limit a data breach, which should be followed by remediation measures. This applies to all data processing components and not only to vNSFs. The Store and any interface requiring login credentials etc. should follow the same approach to secure a data breach. Breach notification is addressed in SHIELD requirements.	Develop/Test/Maintain an incident response plan, a data breach notification system, determine the scope/scale of the breach, notify the Data Protection Officer, investigate the data breach and provide documentation and reports. The Trust Monitor also provides early warning against breaches in vNSFs.
P4 Insufficient Deletion of Personal Data	Appropriate data retention periods should be defined. After the retention period is over the data should be deleted, (or upon request by the data subject). If retention is not necessary, the data should be deleted after processing. Retention is addressed in the specifications following in subsection 4.3.	Follow the data minimization principle and adhere to GDPR data subject rights (e.g right of deletion, restriction of processing, right to be forgotten etc.), document data retention policies, deletion should be verifiable.
P5 Non-transparent policies, terms and conditions	This relates to not providing sufficient information to describe how data is collected, processed, stored, managed etc. This information should always be easily accessible and understandable. SHIELD provides this information in the specifications for each data processing component. The following subsection 4.4 discusses assurance and certification.	Develop terms and conditions for the SHIELD services, make information available and comprehensible, separate terms and conditions for GDPR in a contract, use visual materials (icons, pictograms etc.), document changes to terms and conditions, keep track of user consent, provide opt-out policies (when feasible).
P6 Collection of data not required for the primary purpose	The collection of user-related data that are not necessary for the purposes of the system is a major privacy risk. This applies to data that were collected without the data subject's knowledge or consent. Purpose is addressed in the specifications following in subsection 4.3, while certification/assurance is addressed in 4.5.	The purpose of data collection/processing should be transparent. Data should only be collected for the specified purpose (data reduction/minimization), opt-out policies should be set when feasible, apply conditioned collection (only under specific circumstances)
P7 Sharing of Data with third party	Provision of a user's data to a third party without the user's knowledge and consent. The existence of APIs for third party data	Proxy the content on self-hosted servers and not directly with a third party, apply tokenization or anonymization, develop a monitoring

	exchange should be clear. If data are being monetized the user should be aware. This is addressed in the specifications following in subsection 4.3 (which defines types of third parties), while certification/assurance is addressed in 4.5.	framework that can whitelist/blacklist third parties, develop appropriate contractual arrangements, monitor user complaints, special provisions should be made for cross-border sharing.
P8 Outdated personal data	The use of outdated, incorrect or bogus user data, failure to update or correct the data. In SHIELD this applies particularly to the IPs involved in remediation actions and to the login credentials of the data processors.	Implement a procedure to obtain input from users and update their data, ability to roll back a remediation action in case of false positive attack detection, in case of updates all related subsystems should be aware.
P9 Missing or insufficient Session Expiration	Failure to effectively enforce a session termination. May result in additional data collection without the user's consent or awareness or even to theft of credentials. In SHIELD this applies particularly in the user interfaces requiring login credentials (e.g. administrators, dashboard users etc.)	Automatic session expiration should be set with appropriate expiration times based on the criticality of the application and the data. Session timeout could be configurable, reminder messages to log out can be implemented.
P10 Insecure Data Transfer	Failure to provide data transfers over encrypted and secured channels, may lead to data leaks, failure to limit the leak surface. In SHIELD this applies to the data transfer between vNSF-vNSFO, vNSF-DARE, DARE-vNSFO.	Send personal data through secure protocols, apply secure configurations, allow connections over secure protocols and disallow unsafe connections, avoid inclusion of personal information in session ID/URL, activate privacy extensions (e.g in IPv6)

In terms of the Store, Orchestrator and Trust Monitor, the main GDPR compliance mechanisms involve the use of identifiable data in the form of their administrators' login credentials. In this case, credentials are stored in encrypted form and can be removed from the system if a user or administrator is no longer required to work with these components. Session expiration is also a mitigation measure that minimises risk for a data breach. In terms of the vNSF components, a detailed analysis follows.

4.3. Regulatory compliance specifications

Based on the previous analysis in 4.1-4.2, SHIELD developed a template for the definition of the regulatory compliance specifications for each vNSF, showcased in Tables 3-12. This template can of course be adapted to any components that store or process personal data (internal or external to SHIELD). SHIELD provides instruction on how to fill out the specifications for a component and provides the specifications for all SHIELD Y1-Y2 vNSFs. The information provided in the template is organised in specific sections:

General Information: This includes basic information on the vNSF such as its name, its developer and a brief description of its key function. It also includes any certification or standardisation marks.

Table 3 vNSF general information.

1	General Information	vNSF Name	<vnsf name>
		vNSF version	<vnsf version number>
		vNSF Developer	<vnsf developer>
		vNSF Description	<description of vnsf>
		Certification & Standardisation	<any existing certification or standardisation marks>

Interfaces and Formats: This is a brief overview of all the inputs and outputs that are programmed in the vNSF. This includes all interfaces and a mention of all standard and non-standard data formats.

Table 4 Overview of interfaces and data formats.

2	Interfaces and Formats	Data Inputs	<description of data inputs>
		Data Outputs	<description of data outputs>
		Data Formats	<description of data types and formats>

Data Types (based on Article 4 & Article 11): This section overviews the way that the GDPR applies to the vNSF. It contains information on the types of personal data that can be parsed by the vNSF (e.g if it collects IP addresses, emails, cookies etc.), any data in special categories (e.g. medical, political, religious etc. This generally does not apply to the specific SHIELD vNSFs, although it might be used in future developments). Identifiability refers to the possibility that the data help identify a specific data subject with processing that is **internal** to the vNSF. This helps assess the impact of a data breach and the level of protection that must be applied, within the DPIA. It is the responsibility of the vNSF developer to include which types of personal data can be parsed by the vNSF. An example is IP addresses in L3 network data, HTTP Cookies in L7 Data, etc. An analysis per protocol might be required[2].

Table 5 Data types.

3	Data types	Personal Data	Y/N	<description of personal data types processed>
		Special Categories	Y/N	<description of special/sensitive data processed>
		Identifiability	Y/N/P	<is the data identifiable within the vNSF?>

Data Storage: This section details how the vNSF stores data, what is the retention period, if there are additional protection mechanisms. It is the responsibility of the vNSF developer to apply data protection in the form of encryption/pseudonymisation/anonymisation.

Table 6 Data storage.

4	Data Storage	Data Storage	Y/N	<description of local data storage>
		Data Encryption	Y/N	<description of encryption scheme>
		Data Retention	Y/N	<retention period for data>

[2] (e.g. the headers From, Authorization, Proxy-Authorization, User-Agent, X-ATT-DeviceId, X-Wap-Profile, X-UIDH, X-Csrf-Token, X-Request-ID, X-Correlation-ID, Set-Cookie could lead to identification of a person or device within HTTP traffic).

		Pseudonymisation	Y/N	<are the data decoupled or pseudonymised?>
		Anonymisation	Y/N	<are the data decoupled or anonymised?>

Data Processing: This section details the processing of personal data within the vNSF. It includes purpose, if processing is monetized or profiles the individual, a description of the data processing algorithm, and a description of the obligations of the data processor etc. It includes a justification on the lawfulness of processing and what is considered to be legitimate use for the vNSF.

Table 7 Data processing activities performed by the vNSF.

5	Data Processing	Purpose	<purpose of data processing>	
		Monetisation	Y/N	<are the data being monetized?>
		Profiling	Y/N	< personal aspects relating to a natural person?>
		Data Processing	Y/N	<description of data processing algorithm>
		Data Processor	<who has access to the data & what are the obligations of the controller>	
		Data Protection Officer	<obligations of the Data Protection Officer>	
		Data Controller	<obligations of the Data Controller>	
		Consent processes	<Requirements for consent processes>	
		Lawfulness	<description of the lawful uses of the vNSF>	

Data sharing: This section details the possible data recipients. It lists the APIs and interfaces that are available to the vNSF for data sharing. It considers GDPR stipulations, as well as the needs of law enforcement and cybersecurity agencies. The vNSF provider is responsible to make clear which APIs are available for a data sharing, but the service provider that chooses to on-board the vNSF may opt-out from using them. It is the responsibility of the service provider to provide information to their clients on how their data are being shared and if they are being monetized or re-used.

Table 8 Available APIs/interfaces for data sharing per recipient category.

6	Data sharing	SHIELD components	Y/N	<which SHIELD components get data from the vnsf>
		Third parties	Y/N	<which third parties get data from the vnsf>
		Law enforcement	Y/N	<special API for law enforcement or national CERTs>
		Cross-border sharing	Y/N	<potential for cross border data sharing>
		CERT/CSIRT	Y/N	<access of CERT/CSIRTs to threat information>

Data Subject Rights: This section is relevant if the vNSF retains personal data (such as network flows, IPs etc.). If there is no retention, the data subject rights do not apply. If data are retained but are not identifiable, **Article 11** states that the data subject should provide a way to identify subsets of data relating to them.

Table 9 Data subject rights under the GDPR.

7	Data Subject Rights	Right of access	Y/N	<is there an interface available from the vNSF developer?>
		Right of rectification	Y/N	<is there an interface available from the vNSF developer?>
		Right to be forgotten	Y/N	<is there an interface available from the vNSF developer?>

		Restriction	Y/N	<is there an interface available from the vNSF developer?>
		Notification	Y/N	<does the vNSF generate notifications of a data breach?>
		Data portability	Y/N	<is there an interface available from the vNSF developer to export data from the vNSF?>

Open Internet: This part is relevant to the Open Internet regulation and EU's net neutrality rules. If the vNSF applies traffic classification or rate limiting, it should be justified as lawful according to the regulation's stipulations.

Table 10 Net neutrality specifications.

8	Open Internet	Traffic Classification	Y/N	<Justification for traffic classification>
		Rate Limiting	Y/N	<Justification for rate limiting>

Non-discrimination: This section applies only on vNSFs that perform any sort of behavioural profiling or process data in sensitive categories. In this case, there should be justification of the use of this processing and safeguards should be in place to ensure that the information cannot be misused against the data subject or lead to discriminatory practices of any kind.

Table 11 Non-discrimination and misuse of data.

9	Non-discrimination	Potential for misuse of data	Y/N N/A	<relevant only if data are special category, or if the vnsf profiles the user>

ePrivacy: This section regards processing of communication contents and the identifiability of the data subject. The provider needs to ensure that communications are safe and secure and that no unwarranted processing takes place (with the exception of Lawful Interception).

Table 12 ePrivacy compliance.

10	ePrivacy	Protection of the contents of a communication	Y/N N/A	<relevant only if the vnsf looks into the contents of the communications, i.e. the packet payloads>
		Use of cookies to provide a user experience and track user preferences	Y/N/ N/A	<relevant only if the processing includes cookies or tracks the users preferences>

4.3.1. vNSF Compliance Specifications

According to the template that was presented, SHIELD provides the compliance specifications of the SHIELD vNSFs in Tables 13-19.

Table 13 Compliance specifications for the NCSRD L3 firewall.

1	General Information	vNSF Name	L3 firewall
		vNSF version	V0.1
		vNSF Developer	NCSRD

		vNSF Description	This vNSF applies firewall rules to IP traffic	
		Certification & Standardisation	None	
2	Interfaces and Formats	Data Inputs	Ingress traffic, vNSFO traffic	
		Data Outputs	Egress traffic	
		Data Formats	Rules received by the vNSFO in xml format, netflow traffic	
3	GDPR applicability	Personal Data	Y	IP addresses
		Special Categories	N	Does not apply
		Identifiability	N	No identifiability within the vNSF
4	Data Storage	Data Storage	N	Network traffic is not stored or retained. The firewall only stores firewall rules sent by the vNSFO
		Data Retention	N/A	There is no retention of network data in the vNSF
		Data Encryption	N/A	Does not apply since there is no retention
		Pseudonymisation	N/A	Does not apply
		Anonymisation	N/A	Does not apply
5	Data Processing	Purpose	Cybersecurity	
		Profiling	N	No behavioural profiling of data subjects
		Monetisation	N	Does not apply
		Data Processing	Network data IPs are checked against set firewall rules to block or limit specific types of traffic (based on IP, protocol, port etc.)	
		Data Processor	The administrator of the vNSF can act as the data processor.	
		Data Protection Officer	Information must be available to the data subject. A Data Protection Officer must be appointed by the data controller.	
		Data Controller	The service provider that deploys the vNSF is the data controller. In case of SecaaS deployments, joint controllers may be envisioned.	
		Consent processes	The client should be informed if the firewall is running on the network and consent to its use. The Data Controller is obligated to obtain consent.	
		Lawfulness	Lawful use of the firewall includes cybersecurity, to ensure infrastructure security and resilience and to protect the interests of the ISP clients.	
6	Data sharing	Other SHIELD components	Y	The vNSF can capture data and send them to DARE for further analysis.
		Third parties	N	No APIs for third party access.
		Law enforcement	N	No APIs for Law Enforcement access; events and mitigation actions can be reviewed through the dashboard.
		Cross-border data sharing	N	No APIs for cross-border data sharing.
		CERT/CSIRT	N	No APIs for CERT/CSIRT access; events and mitigation actions can be reviewed through the dashboard.
7	Data Subject Rights	Right of access	Y	There is no identifiability and no data retention with respect to network traffic, hence other measures to ensure the data subject rights do not apply. The IP addresses stored in iptables rules can be removed through the dashboard or directly via the vNSF management interface.
		Right of rectification		
		Right to be forgotten		
		Restriction		
		Notification		
		Data portability		
8	Open Internet	Traffic Classification	N	No traffic classification.
		Rate Limiting	Y	vNSF can apply rate limiting rules per protocol or IP etc. for cyberattack mitigation.

9	Non Discrimination	Potential for misuse	N/A	The vNSF does not profile the user or process data in sensitive categories, hence the potential for misuse is minimized.
10	ePrivacy	Protection of the contents of a communication	N/A	The vNSF does not process the contents of communications.
		Use of cookies to provide a user experience and track user preferences	N/A	The vNSF does not utilise cookies in any form.

Table 14 vIDS compliance specifications.

1	General Information	vNSF Name	vIDS (Virtual Intrusion Detection System)	
		vNSF version	v0.1	
		vNSF Developer	NCSRD	
		vNSF Description	This vNSF monitors and logs the network traffic for signs of malicious activity and generates an alert upon discovery of a suspicious event.	
		Certification & Standardisation	None	
2	Interfaces and Formats	Data Inputs	Ingress traffic, vNSFO traffic (rules configuration)	
		Data Outputs	Egress traffic (IPS mode), output (alerts) on threat identification that includes specific threat information	
		Data Formats	Rules received by the vNSFO (security orchestrator) in JSON format, IDS alerts stored are Unified 2 (u2) format, IDS alerts sent to DARE are in JSON.	
3	GDPR applicability	Personal Data	Y	IP addresses, packet headers may include emails, http cookies etc.
		Special Categories	N	Does not apply.
		Identifiability	Y	Certain packet headers could provide identifiable information (e.g. emails, login credentials etc.).
4	Data Storage	Data Storage	N	Network traffic is not stored or retained after processing. Alerts with information of the identified threats are the output of the vNSF which are stored in u2 format (IDS logs). Alerts are transformed to JSON to be sent to DARE but are not permanently stored.
		Data Retention	Y	vIDS alerts are stored both in U2 and JSON formats and vIDS logs are stored in U2 format. The alerts might contain information of source/destination IP addresses, source/destination ports, protocols used, rate of packets per specified unit of time, Perl Compatible Regular Expressions (PCREs), content contained in the packet payload that might also include sensitive information such as personal data, urls, telephone numbers, e-mail addresses, etc.
		Data Encryption	N	No data encryption in this vIDS version.
		Pseudonymisation	N	The vNSF does not pseudonymise traffic.
		Anonymisation	N	The vnSF does not anonymise traffic.
5	Data Processing	Purpose	Cybersecurity monitoring and reporting an incident to a CERT.	
		Profiling	N	No behavioural profiling of data subjects.
		Monetisation	N	Does not apply.

		Data Processing		This vNSF uses signatures/rules to identify security threats and possible attacks based on traffic patterns or specific information contained in the traffic.
		Data Processor		The administrator of the vNSF can act as the data processor.
		Data Protection Officer		Information must be available to the data subject. A Data Protection Officer must be appointed by the data controller.
		Data Controller		The service provider that deploys the vNSF is the data controller. In case of SecaaS deployments, joint controllers may be envisioned.
		Consent processes		The client should be informed if the vNSF is running on the network and consent to its use.
		Lawfulness		Lawful use of the vIDS includes cybersecurity monitoring in case of an incident, the vNSF monitors the offending traffic and exports statistics in STIX 2.0 format for a CERT or other national cybersecurity agency.
6	Data sharing	Other SHIELD components	Y	vIDS alerts are transformed to JSON and are sent to DARE.
		Third parties	N	No APIs for third party access.
		Law enforcement	N	No APIs for Law Enforcement access; events and mitigation actions can be reviewed through the dashboard or through external visualization tools.
		Cross-border data sharing	N	No APIs for cross-border data sharing.
		CERT/CSIRT	Y	CERT/CSIRTs can receive information in STIX 2.0 format.
7	Data Subject Rights	Right of access	Y	There is no identifiability and no data retention with respect to network traffic, hence other measures to ensure the data subject rights do not apply. The IP addresses stored in rules or logs can be removed through the dashboard or directly via the vNSF management interface.
		Right of rectification		
		Right to be forgotten		
		Restriction		
		Notification		
		Data portability		
8	Open Internet	Traffic Monitoring	Y	The vNSF monitors traffic and identifies specific patterns that signify a possible security threat. Use for cybersecurity monitoring is legitimate according to the Open Internet Regulation.
		Rate Limiting	N	The vNSF does not apply rate limiting rules, does not discriminate against specific types of traffic.
9	Non Discrimination	Potential for misuse	N	The vNSF does not profile the user or process data in sensitive categories, hence the potential for misuse is minimized.
10	ePrivacy	Protection of the contents of a communication	N/A	The vNSF does not process the contents of communications.
		Use of cookies to provide a user experience and track user preferences	N	The vNSF does not utilise cookies. Cookies may be part of the network traffic in higher OSI layers and receive the same protection as other personal data under the GDPR.

Table 15 vDPI compliance specifications.

1	General Information	vNSF Name	vDPI (Virtual Deep Packet Inspection)	
		vNSF version	v0.4	
		vNSF Developer	ORION	
		vNSF Description	This vNSF inspects packet headers and classifies traffic in categories (e.g. per application type etc.)	
		Certification & Standardisation	None	
2	Interfaces and Formats	Data Inputs	Ingress traffic, vNSFO traffic	
		Data Outputs	Egress traffic, classification results/statistics, STIX threat information	
		Data Formats	Rules received by the vNSFO in xml format, netflow traffic	
3	GDPR applicability	Personal Data	Y	IP addresses, packet headers may include emails, http cookies etc.
		Special Categories	N	Does not apply
		Identifiability	Y	Certain packet headers could provide identifiable information (e.g. emails, login credentials etc.)
4	Data Storage	Data Storage	N	Network traffic is not stored or retained after processing, only classified. Results of the classification are output from the vNSF.
		Data Retention	N	Retention ends after processing
		Data Encryption	N	No data encryption in this vDPI version
		Pseudonymisation	N	The vNSF does not pseudonymise traffic
		Anonymisation	N	The vnSF does not anonymise traffic
5	Data Processing	Purpose	Cybersecurity monitoring and reporting an incident to a CERT	
		Profiling	N	No behavioural profiling of data subjects
		Monetisation	N	Does not apply
		Data Processing	This vNSF uses nDPI to classify traffic types.	
		Data Processor	The administrator of the vNSF can act as the data processor.	
		Data Protection Officer	Information must be available to the data subject. A Data Protection Officer must be appointed by the data controller.	
		Data Controller	The service provider that deploys the vNSF is the data controller. In case of SecaaS deployments, joint controllers may be envisioned.	
		Consent processes	The client should be informed if the vNSF is running on the network and consent to its use.	
		Lawfulness	Lawful use of the vDPI includes cybersecurity monitoring in case of an incident, the vNSF monitors the offending traffic and exports statistics in STIX 2.0 format for a CERT or other national cybersecurity agency.	
6	Data sharing	Other SHIELD components	Y	The vNSF may send classification results to an InFlux DB which are visualized in a dashboard.
		Third parties	N	No APIs for third party access
		Law enforcement	N	No APIs for Law Enforcement access; events and mitigation actions can be reviewed through the dashboard or through external visualization tools.
		Cross-border data sharing	N	No APIs for cross-border data sharing
		CERT/CSIRT	Y	CERT/CSIRTs can receive information in STIX 2.0 format
7	Data Subject Rights	Right of access	N/A	There is no data retention once processing ends; hence the data subject rights measures do not apply.
		Right of rectification		
		Right to be forgotten		
		Restriction		
		Notification		
		Data portability		

8	Open Internet	Traffic Classification	Y	The vNSF monitors and classifies traffic. Use for cybersecurity monitoring or load balancing is legitimate according to the Open Internet Regulation.
		Rate Limiting	N	The vNSF does not apply rate limiting rules, does not discriminate against specific types of traffic
9	Non Discrimination	Potential for misuse	N	The vNSF does not profile the user or process data in sensitive categories, hence the potential for misuse is minimized.
10	ePrivacy	Protection of the contents of a communication	N/A	The vNSF does not process the contents of communications.
		Use of cookies to provide a user experience and track user preferences	N	The vNSF does not utilise cookies. Cookies may be part of the network traffic in higher OSI layers and receive the same protection as other personal data under the GDPR.

Table 16 ProxyTLS compliance specifications.

1	General Information	vNSF Name	ProxyTLS	
		vNSF version	v0.1	
		vNSF Developer	TID	
		vNSF Description	This vNSF proxy HTTP/S traffic to detect and block malicious URLs.	
		Certification & Standardisation	None	
2	Interfaces and Formats	Data Inputs	Ingress traffic, vNSFO traffic, Mgmt traffic	
		Data Outputs	Egress traffic, security proxy logs to DARE	
		Data Formats	Rules received by the vNSFO in xml format, text based logs	
3	GDPR applicability	Personal Data	Y	IP addresses, HTTP packet headers may include user agents and destination URLs
		Special Categories	N	Does not apply
		Identifiability	Y	Certain http packet headers could provide identifiable information (e.g. login credentials in url.)
4	Data Storage	Data Storage	N	Network traffic is not stored or retained after processing, only detection. Results of the processing are output from the vNSF.
		Data Retention	N	Retention ends after processing
		Data Encryption	N	No data is encrypted
		Pseudonymisation	N	The vNSF does not pseudonymise traffic
		Anonymisation	N	The vNSF does not anonymise traffic
5	Data Processing	Purpose	Cybersecurity monitoring and reporting an incident to a CERT	
		Profiling	N	No behavioural profiling of data subjects
		Monetisation	N	Does not apply
		Data Processing	HTTP headers URL field is collected and compare with a blacklist. Matchs are reported to DARE and in case of remediation the flow is stopped. In the case of HTTPS the process is similar but TLS session is end and reconstructed to read the header.	
		Data Processor	The administrator of the vNSF can act as the data processor.	
		Data Protection Officer	Information must be available to the data subject. A Data Protection Officer must be appointed by the data controller.	

		Data Controller	The service provider that deploys the vNSF is the data controller. In case of SecaaS deployments, joint controllers may be envisioned.	
		Consent processes	The client should be informed that his HTTP traffic is inspected and decrypted by security reason. Indeed, explicit consents is necessary for use the functionality, because clients need to connect and install a ProxyTLS certificate.	
		Lawfulness	Lawful use of the HTTP monitor includes cybersecurity monitoring and traffic filtering in case of an incident.	
6	Data sharing	Other SHIELD components	Y	The vNSF may send all proxy traffic logs not only malicious ones based on blacklist for ML analysis in DARE.
		Third parties	N	No APIs for third party access
		Law enforcement	N	No APIs for Law Enforcement access; events and mitigation actions can be reviewed through the SHIELD dashboard
		Cross-border data sharing	N	No APIs for cross-border data sharing
		CERT/CSIRT	N	No APIs fro CERT
7	Data Subject Rights	Right of access	N/A	There is no data retention once processing ends; hence the data subject rights measures do not apply.
		Right of rectification		
		Right to be forgotten		
		Restriction		
		Notification		
		Data portability		
8	Open Internet	Traffic Classification	Y	The vNSF monitors and classifies traffic as malicious or benign. Use for cybersecurity monitoring is legitimate according to the Open Internet Regulation.
		Rate Limiting	N	The vNSF does not apply rate limiting rules, does not discriminate against specific types of traffic
9	Non Discrimination	Potential for misuse	N	The vNSF does not profile the user or process data in sensitive categories, hence the potential for misuse is minimized.
10	ePrivacy	Protection of the contents of a communication	N/A	The vNSF does not process the contents of communications. Only work with HTTP headers.
		Use of cookies to provide a user experience and track user preferences	N	The vNSF does not process cookies. Cookies may be part of the network traffic in HTTP header or body but these are ignored.

Table 17 HTTP/S Analyser compliance specifications.

1	General Information	vNSF Name	HTTP/S Analyser	
		vNSF version	v0.1	
		vNSF Developer	TID	
		vNSF Description	This vNSF generate network flows and classify traffic without content analysis.	
		Certification & Standardisation	None	
2	Interfaces and Formats	Data Inputs	Ingress traffic (Mirrored), vNSFO traffic, Mgmt traffic	
		Data Outputs	Network flows and labels of classification to DARE	
		Data Formats	Rules received by the vNSFO in xml format, netflow, log text file	
3		Personal Data	Y	IP addresses

	GDPR applicability	Special Categories	N	Does not apply.
		Identifiability	N	Only by ISP if correlate IPs with identities, i.e.: subscription database and IPs dynamic assignment. Does not provide identifiability within the vNSF.
4	Data Storage	Data Storage	N	Network traffic is not stored or retained after processing, only detection. Results of the processing are the output from the vNSF.
		Data Retention	N	Retention ends after processing.
		Data Encryption	N	No data is encrypted.
		Pseudonymisation	N	The vNSF does not pseudonymise traffic.
		Anonymisation	N	The vNSF does not anonymise traffic.
5	Data Processing	Purpose	Cybersecurity classification in a family of the traffic	
		Profiling	N	No behavioural profiling of data subjects
		Monetisation	N	Does not apply
		Data Processing	Layer 3-4 data analysis to estimate a label where to classify the traffic. The possible labels are: web. Video, storage, other.	
		Data Processor	The administrator of the vNSF can act as the data processor.	
		Data Protection Officer	Information must be available to the data subject. A Data Protection Officer must be appointed by the data controller.	
		Data Controller	The service provider that deploys the vNSF is the data controller. In case of SecaaS deployments, joint controllers may be envisioned.	
		Consent processes	The client should be informed if the vNSF is running on the network and consent to its use.	
		Lawfulness	Lawful use of the HTTP monitor includes cybersecurity monitoring and traffic classification in case of an incident.	
6	Data sharing	Other SHIELD components	Y	The vNSF may send classified traffic flow logs to DARE.
		Third parties	N	No APIs for third party access
		Law enforcement	N	No APIs for Law Enforcement access; events and mitigation actions can be reviewed through the SHIELD dashboard
		Cross-border data sharing	N	No APIs for cross-border data sharing
		CERT/CSIRT	N	No APIs for CERT
7	Data Subject Rights	Right of access	N/A	There is no data retention once processing ends; hence the data subject rights measures do not apply.
		Right of rectification		
		Right to be forgotten		
		Restriction		
		Notification		
		Data portability		
8	Open Internet	Traffic Classification	Y	The vNSF monitors and classifies traffic in generic categories. Use for cybersecurity monitoring is legitimate according to the Open Internet Regulation.
		Rate Limiting	N	The vNSF does not apply rate limiting rules, does not discriminate against specific types of traffic
9	Non Discrimination	Potential for misuse	N	The vNSF does not profile the user or process data in sensitive categories, hence the potential for misuse is minimized.
10	ePrivacy	Protection of the contents of a communication	N/A	The vNSF does not process the contents of communications.

		Use of cookies to provide a user experience and track user preferences	N	The vNSF does not process cookies.

Table 18 - Compliance specifications for the POLITO L3 Filter.

1	General Information	vNSF Name	L3 Filter	
		vNSF version	V0.1	
		vNSF Developer	POLITO	
		vNSF Description	This vNSF applies filtering rules to network traffic depending on the (source, destination) IP address and port	
		Certification & Standardisation	None	
2	Interfaces and Formats	Data Inputs	Ingress traffic, vNSFO traffic	
		Data Outputs	Egress traffic	
		Data Formats	Rules received by the vNSFO in xml format, netflow traffic	
3	GDPR applicability	Personal Data	Y	IP addresses
		Special Categories	N	Does not apply
		Identifiability	N	No identifiability within the vNSF
4	Data Storage	Data Storage	N	Only firewall rules are stored by the vNSF, not actual traffic.
		Data Retention	N/A	There is no retention of network data in the vNSF
		Data Encryption	N/A	Does not apply since there is no retention
		Pseudonymisation	N/A	Does not apply
		Anonymisation	N/A	Does not apply
5	Data Processing	Purpose	Cybersecurity/Filtering	
		Profiling	N	No behavioural profiling of data subjects
		Monetisation	N	Does not apply
		Data Processing	Network data IPs are checked against set firewall rules to block or limit specific types of traffic (based on IP, protocol, port etc.)	
		Data Processor	The administrator of the vNSF can act as the data processor.	
		Data Protection Officer	Information must be available to the data subject. A Data Protection Officer must be appointed by the data controller.	
		Data Controller	The service provider that deploys the vNSF is the data controller. In case of SecaaS deployments, joint controllers may be envisioned.	
		Consent processes	The client should be informed if the firewall is running on the network and consent to its use.	
		Lawfulness	Lawful use of the firewall includes cybersecurity, to ensure infrastructure security and resilience and to protect the interests of the ISP clients.	
6	Data sharing	Other SHIELD components	Y	The vNSF can capture data and send them to DARE for further analysis (via distributed collector)
		Third parties	N	No APIs for third party access
		Law enforcement	N	No APIs for Law Enforcement access; events and mitigation actions can be reviewed through the dashboard
		Cross-border data sharing	N	No APIs for cross-border data sharing
		CERT/CSIRT	N	No APIs for CERT/CSIRT access; events and mitigation actions can be reviewed through the dashboard
7	Data Subject Rights	Right of access	N/A	
		Right of rectification		

		Right to be forgotten		There is no identifiability and no data retention hence other measures to ensure the data subject rights do not apply.
		Restriction		
		Notification		
		Data portability		
8	Open Internet	Traffic Classification	N	No traffic classification.
		Rate Limiting	Y	vNSF can apply rate limiting rules per protocol or IP (specific, range) for cyberattack mitigation.
9	Non Discrimination	Potential for misuse	N/A	The vNSF does not profile the user or process data in sensitive categories, hence the potential for misuse is minimized.
10	ePrivacy	Protection of the contents of a communication	N/A	The vNSF does not process the contents of communications, only headers.
		Use of cookies to provide a user experience and track user preferences	N/A	The vNSF does not track user experience and it does not store cookies.

Table 19 - Compliance specifications for the POLITO Forward L7 Filter

1	General Information	vNSF Name	Forward L7 Filter	
		vNSF version	V0.1	
		vNSF Developer	POLITO	
		vNSF Description	This vNSF inspects traffic for specific Layer 7 protocols and headers (e.g. HTTP, FTP), URL filtering and Access Control List as a reverse proxy and Web Application Firewall.	
		Certification & Standardisation	None	
2	Interfaces and Formats	Data Inputs	Ingress traffic, vNSFO traffic	
		Data Outputs	Egress traffic, classification results	
		Data Formats	Rules received by the vNSFO in xml format, netflow traffic	
3	GDPR applicability	Personal Data	Y	IP addresses, packet headers may include emails, http cookies etc.
		Special Categories	N	Does not apply
		Identifiability	Y	Certain packet headers could provide identifiable information (e.g. emails, login credentials etc.).
4	Data Storage	Data Storage	N	Network traffic is not stored or retained after processing.
		Data Retention	Y	Retention ends after processing.
		Data Encryption	N	Does not apply since there is no long-term retention
		Pseudonymisation	N/A	Does not apply
		Anonymisation	N/A	Does not apply
5	Data Processing	Purpose	Cybersecurity/Monitoring/Filtering	
		Profiling	N	No behavioural profiling of data subjects
		Monetisation	N	Does not apply
		Data Processing	This vNSF uses Mod Security to process traffic headers in a rule-based approach.	
		Data Processor	The administrator of the vNSF can act as the data processor.	
		Data Protection Officer	Information must be available to the data subject. A Data Protection Officer must be appointed by the data controller.	

		Data Controller		The service provider that deploys the vNSF is the data controller. In case of SecaaS deployments, joint controllers may be envisioned.
		Consent processes		The client should be informed if the vNSF is running on the network and consent to its use. It is the responsibility of the service provider to obtain consent.
		Lawfulness		Lawful use of the Forward L7 Filter includes cybersecurity monitoring, to ensure infrastructure security and resilience and to protect the interests of the ISP clients.
6	Data sharing	Other SHIELD components	Y	The vNSF can capture data and send them to DARE for further analysis (via distributed collector). Data may include logged requests detected by the Web Application Firewall (in case they match any of the filtering rules).
		Third parties	N	No APIs for third party access
		Law enforcement	N	No APIs for Law Enforcement access; events and mitigation actions can be reviewed through the dashboard
		Cross-border data sharing	N	No APIs for cross-border data sharing
		CERT/CSIRT	N	No APIs for CERT/CSIRT access; events and mitigation actions can be reviewed through the dashboard
7	Data Subject Rights	Right of access	Y	There is no data retention once processing ends, hence other measures to ensure the data subject rights do not apply. Regarding firewall rules, they can be altered via the dashboard or the vNSF management interface upon request to the data processor.
		Right of rectification		
		Right to be forgotten		
		Restriction		
		Notification		
		Data portability		
8	Open Internet	Traffic Classification	Y	The vNSF monitors and classifies traffic with a rule-based approach. Use for cybersecurity monitoring or load balancing is legitimate according to the Open Internet Regulation.
		Rate Limiting	N	The vNSF does not apply rate limiting rules, does not discriminate against specific types of traffic
9	Non Discrimination	Potential for misuse	N/A	The vNSF does not profile the user or process data in sensitive categories, hence the potential for misuse is minimized.
10	ePrivacy	Protection of the contents of a communication	Y	The vNSF can be instructed to process request and response payloads along their headers. This capability is required to address web-based attacks that are pursued by including specific information in the payload. No data is used to eavesdrop the user. The vNSF can inspect payloads of packets, but no data is used to profile the user. The vNSF does not inspect encrypted traffic. The vNSF is not capable of monitoring messaging, personal communications, emails etc.
		Use of cookies to provide a user experience and track user preferences	N	No use of cookies.

Based on the specifications provided by each vNSF developer, SHIELD can trace the vNSF to the ERC requirements they need to fulfil (Table 20).

Table 20 ERC requirements tracing.

SHIELD vNSF	ERC requirement	Description
vIDPS	ERC01 Access to and portability of personal data, ERC02 Data rectification and erasure, ERC03 Access to related Data Protection information	Access is provided by an external interface as well as via the vNSF management interface.
vDPI	ERC01 Access to and portability of personal data, ERC02 Data rectification and erasure, ERC03 Access to related Data Protection information	Access is provided by an external interface as well as via the vNSF management interface.
	ERC06 Transparency in traffic classification, ERC08 Net Neutrality	vDPI classifies traffic but does not take remedial actions or throttle the rate.
ProxyTLS	ERC04 Trasparency in data processiong, ERC6 Transparency in traffic classification, ERC08 Net Neutrality	Classification between malicious or bening traffic is done based on known blacklist or provided by the ISP, traffic filtering is based on security reasons.
	ERC01 Access to and portability of personal data, ERC02 Data rectification and erasure, ERC03 Access to related Data Protection information	No personal data is retained to be portable, modified or erased in the vNSF. Access to Data protection information will be done through the SHIELD Dashboard. i.e. DARE data retained
HTTP/S analyser	ERC11 Privacy and Security by design, ERC12 ePrivacy	Classification process is design to use only Layer 3-4, data therefore no payload is analyzed. This is a clear effort to preserve the user privacy on the communication content in the vNSF desing.
	ERC04 Transparency in data processiong, ERC06 Transparency in traffic classification, ERC08 Net Neutrality	Classification is done by a automate process to assign a label, based in machine learning techniques, not modification or alteration is done in the labeling or in the traffic to bias the process.
	ERC01 Access to and portability of personal data, ERC02 Data rectification and erasure, ERC03 Access to related Data Protection information	No personal data is retained to be portable, modified or erased in the vNSF. Access to Data protection information will be done through the SHIELD Dashboard. i.e. DARE data retained
L3 Filter & Y1 vFW	ERC01 Access to and portability of personal data, ERC02 Data rectification and erasure, ERC05 Data retention	The vNSF does not retain personal data and does not allow data to be modified or erased in the vNSF.

	ERC09 Lawful Interception	The vNSF does not change public IP addresses or encrypts traffic, hence it does not require to integrate a LI system.
Forward l7 Filter	ERC01 Access to and portability of personal data, ERC02 Data rectification and erasure, ERC05 Data retention	The vNSF only retains personal data for the processing. After processing, data is automatically removed. No data can be modified in the vNSF.

4.3.2. Integrating specifications in the Store

The legal compliance specifications, as well as the technical specifications for the vNSFs should be visible in the Store. This would enable the user to select which vNSFs to use by making an informed decision. In the context of privacy and data protection it is considered a good practice to use privacy iconsets that allow the user to quickly visualize data protection concepts without legal and technical jargon. An example of an iconset is presented in Figure 25. Icons include the type of data that are used, the various types of processing, how it is stored and shared, or deleted. This concept can easily be adapted for SHIELD, although a more comprehensive iconset would be necessary to cover the breadth of the compliance specifications. Figure 26 provides a mockup for the NCSRD L3 filter that was utilised in Y1 demos. This way, the user browsing the store can view the compliance specifications for the vNSFs and make an informed decision on which vNSF to utilise. Access to the full specifications (compliance and technical) is of course, also available.

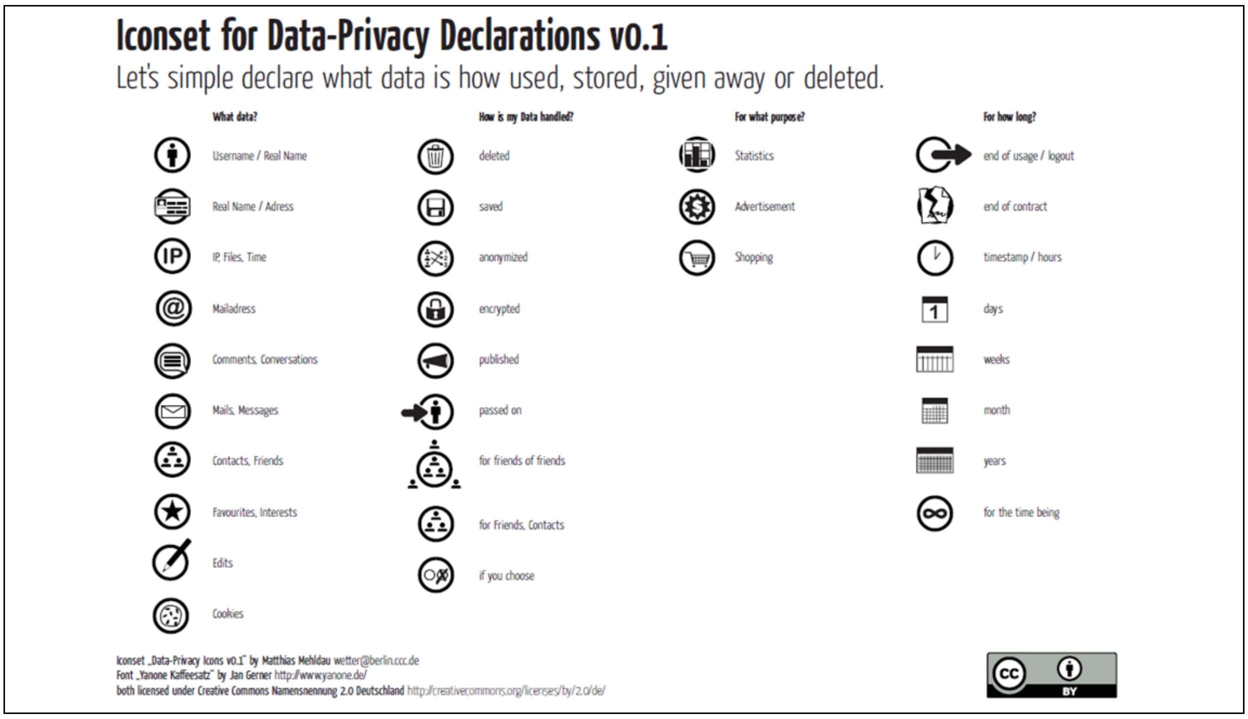
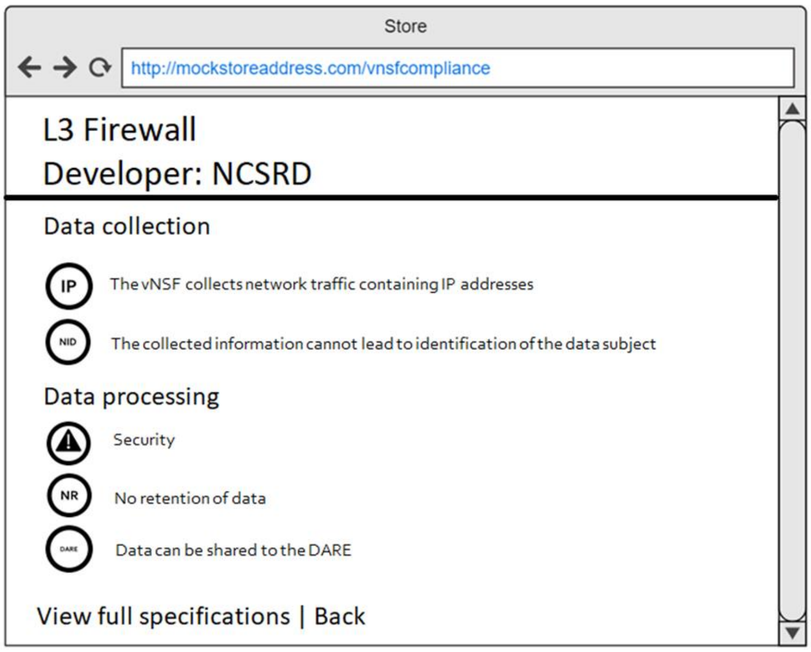


Figure 25: Iconset for Data-Privacy Declarations v0.1 (by Matthias Mehldau, licensed under Creative Commons 2.0).


Store
http://mockstoreaddress.com/vnsfcompliance
IP
NID
NR
DARE
View full specifications | Back

L3 Firewall
Developer: NCSRD

Data collection

The vNSF collects network traffic containing IP addresses

The collected information cannot lead to identification of the data subject

Data processing

Security

No retention of data

Data can be shared to the DARE

Figure 26: Mock-up showing how to the compliance specifications can be easily visualised in the Store.

4.4. Compliance and GDPR Certification

An important aspect to address at this point is the difference between compliance and certification. Some operational examples of successful privacy certifications include:

- **PrivacyTrust** (formerly eTrust) [70]: a private company that provides privacy certifications for websites and online businesses. A PrivacyTrust certification indicates that a website has been reviewed by the company and is aligned with their privacy and data protection requirements. Similar certifications are available by other providers such as WebTrust, etc. although they are not focused on GDPR or software-oriented architectures
- The **Health Information Trust Alliance** (HITRUST) [71] is a US-based association of organisations, that certifies products for compliance with the Health Insurance Portability and Accountability Act of 1996 [72] (HIPAA). HIPAA sets rules for the handling of medical data in the US. HITRUST, therefore, is able to certify products for HIPAA compliance. It is therefore a case where legal compliance for data protection is certified by an appropriate body, although its scope is not as broad as the EU GDPR.

In SHIELD's case**, certification requires the existence of a trusted third party that inspects the vNSF and verifies that it is compliant with GDPR** and that the information provided in its specifications are accurate. According to the GDPR **Article 42,** *"the Member States, the supervisory authorities, the Board and the Commission shall encourage the establishment of data protection certification mechanisms and of data protection seals and marks".* The certification should be voluntary and transparent, and the certification body should be granted cooperation and access to the processing. **Article 43** of the GDPR states **that certification bodies should be accredited** (ISO 17065). As GDPR is being implemented in each Member State, it is expected that multiple data protection certification providers will be accredited with the relevant national authorities. Thus, it will be possible in the future for vNSF developers to get their products certified for GDPR compliance. At consortium level, SHIELD partners intend to

reach out to certification bodies that are accredited for GDPR compliance certifications, but it is uncertain if certifications will be available across all member states by the date of the project's completion (February 2018).
Apart from the GDPR, compliance with well-known standards and privacy reference frameworks can be taken into account. **ISO/IEC 27001 Information Security Management Systems** [73] is a well-known international standard for information security that provides a set of standardised requirements for an information security management system (ISMS). **ISO/IEC 27018** [74] similarly defines guidelines for implementing personal data protections and specifies controls within ISO/IEC 27001. ISO certification in this context can be considered suitable for SHIELD's case. **ISO/IEC 29100** [75] provides a privacy framework. The **OASIS Privacy Management Reference Model** [76] can also be considered, as an open standard for privacy and data protection.

5. VALIDATION AND TECHNICAL CERTIFICATION

Following the publication of the D5.1 deliverable [13], this section briefly discusses the certification and validation of the SHIELD vNSF ecosystem. This focuses on the certifying the vNSF ecosystem and its performance as carrier-grade (as opposed to GDPR certification or other privacy and data protection certifications).

5.1. Technical certification of vNSFs

Technical certification of vNSFs can be a major step towards their adoption in the market as it increases trust in the final product. Especially in the case of the SHIELD Store, a vNSF that is certified by an appropriate, accredited certification body and bears a certification mark, could potentially be more easily adopted by a client, and at the same time foster the development of the Store into a viable marketplace.

Related certification environments are starting to be available to VNF developers. In most cases, the certification they offer is driven by either a major association of suppliers, or by telco/service providers. The purpose is each case is quite different; telco providers often focus on the certification of VNFs they intend to deploy on their own networks, while suppliers usually focus on the creation of a VNF marketplace based on their cloud offerings.

The Nokia CloudBand ecosystem [77] (Figure 27) is such an example. CloudBand is being marketed as “the world’s first carrier-grade OpenStack NFV platform”. Nokia offers membership to an association of members that comprises VNF developers, other suppliers etc. and allows them to certify their products in this ecosystem. According to Nokia, the purpose of this effort is:

- To validate VNF suppliers’ compliance with NFV standards,
- Expedite deployment and reduce risk,
- Accelerate time to market with pre-validated VNFs,
- Access a broader range of pre-validated VNFs, offering greater choice for service creation,
- Gain the opportunity to create a self-service marketplace for applications for enterprise and residential customers.

The environment is based on OpenStack and the CloudBand line of products which provide the functionalities of the Orchestrator (NFVO), the Virtual Infrastructure Manager (VIM), etc. The creation of a marketplace is aligned with SHIELD’s vision although it is highly dependent on the CloudBand product line. This means that when an NFV supplier is certified for this platform, they are inevitably “locked-in” to use CloudBand products for the deployment of their VNFs. The certification and the association of members, however, has built a “critical mass” since it is backed by a major supplier.

Huawei follows a similar approach in their Open Labs [78]. Open Lab partners include China Mobile, VMware, Red Hat, Canonical and the Linux Foundation, OPNFV etc. These efforts mark a key step to realizing Huawei's future-oriented open SoftCOM architecture, based on their proprietary All Cloud technology.

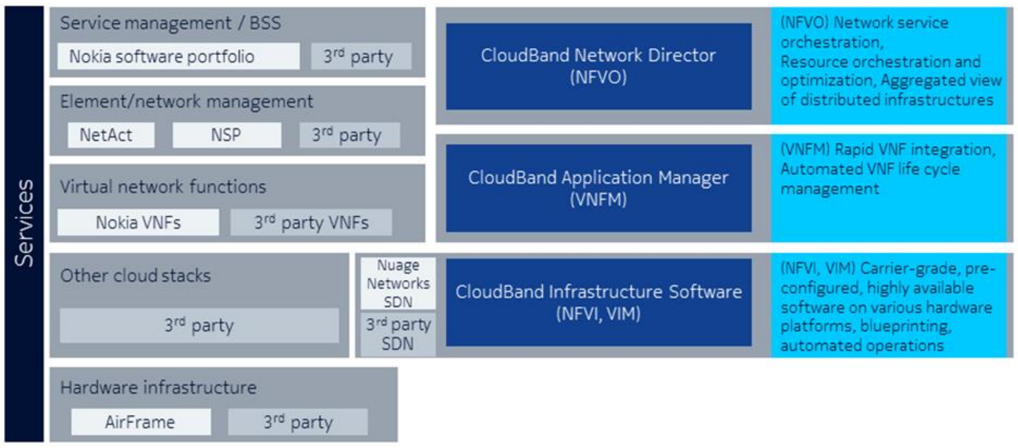

Services
Service management / BSS
Nokia software portfolio
3rd party
Element/network management
NetAct
NSP
3rd party
Virtual network functions
Nokia VNFs
3rd party VNFs
Other cloud stacks
3rd party
Nuage Networks SDN
3rd party SDN
Hardware infrastructure
AirFrame
3rd party
CloudBand Network Director (NFVO)
(NFVO) Network service orchestration, Resource orchestration and optimization, Aggregated view of distributed infrastructures
CloudBand Application Manager (VNFM)
(VNFM) Rapid VNF integration, Automated VNF life cycle management
CloudBand Infrastructure Software (NFVI, VIM)
(NFVI, VIM) Carrier-grade, pre-configured, highly available software on various hardware platforms, blueprinting, automated operations

Figure 27 Nokia's CloudBand certification environment.

Red Hat's VNF certification [79] (Figure 28) can offer a deeper integration for NFV and verify that providers of virtual network functions are taking full advantage of the cloud platform for customer deployments. It is currently used by Cisco, Huawei, Citrix, Ericsson, Juniper networks etc. Red Hat's environment closely resembles the SHIELD environment. It uses the CloudForms Open Source VIM, and a very similar NFVI (e.g. support for KVM and Docker, OpenDayLight SDN Controller, CentOS is closely related to Red Hat Enterprise Linux, etc). RedHat's environment also takes into account VNF standardisation and differs in that it provides validation per vertical industry and for different over-the-top (OTT) services.

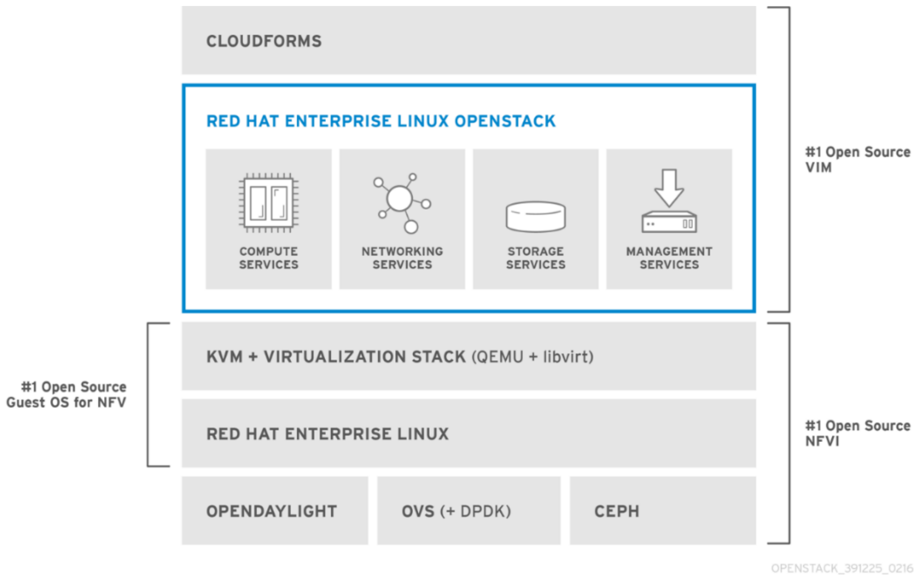


Figure 28 Red Hat's NFV certification environment.

Considering these alternatives, the Red Hat environment is a closer match to SHIELD's architecture and could be considered as a more appropriate selection for the certification of vNSFs for future exploitation. In all three cases, however, the NFVO/VIM used is different from SHIELD which makes exploitation of these certifications within the context of the Store difficult at the time, although it can be easily considered for future deployments. Hence, the best-case scenario for the Store, would be to certify VNFs in an OpenStack/OSM environment similar to SHIELD's current deployment. One such certification environment is supported by **Telefónica's NFV Reference Lab** [80]. As a telco operator, Telefónica uses this certification environment to validate VNFs for operational use on its network. This environment lacks the broader exploitation potential of the previous examples, although it is closer to SHIELD's deployment configuration. Furthermore, Telefónica collaborates closely with ETSI and OpenMANO and supports the development of the OSM Orchestrator. Through partner TID, SHIELD reached out for some information on how a VNF can be certified for carrier-grade performance. Although this information is proprietary, some base requirements were communicated to SHIELD, and were adapted to its architecture.

According to Telefónica, the objective of this certification would be to validate if a vNSF works as intended and without adverse effects to the rest of the SHIELD platform. The test process must be simple and effective and, to result in a certification, it must cover (Figure 29):

- **Documentation:** The vNSF should include a clear deployment guide.
- **Clear Resource Requirements:** The resources that are needed (memory, hard disk, network interfaces etc.) and the requirements should be clearly documented.
- **Instantiation:** the vNSF can be created (and removed) successfully through the orchestrator (OSM) with the Network Service Descriptor (NSD) defined for the tests.
- **Management:** The vNSF management interfaces should be clearly defined and accessible by OSM. Same principle also applies to the policy configurations that originate from the DARE and are relayed to the vNSF through the Orchestrator.
- **Non-functional Testing procedures for vNSF**:
  - **Durability:** Continuous running for periods of 4hs, 8hs, 24hs, 48hs.
  - **Failure recovery:** show report incident and recovery process in the case of VM/platform restart or shutdown,
  - **vNSF hardening** as defined in NF09.
- **Functional Testing procedures for vNSF:**
  - Testing of actual vNSF functionality (as defined in the vNSF documentation).
  - Testing the API with DARE for sending logs and the API for receiving mitigation actions.
  - Testing of vNSF performance: Clearly defined throughput targets are met by the vNSF. Testing includes different packet sizes (including IMIX) and protocols (UDP and TCP). For each target, **delay, packet loss and jitter** should be measured.

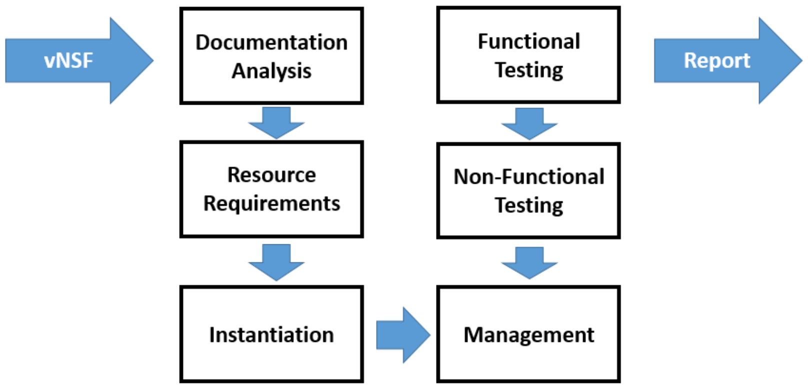


Figure 29 Test process for a vNSF certification.

5.2. Integration and validation tests

The SHIELD test plan that is defined in D5.1 [13] (Section 3 and Annex B Definition of tests) already covers multiple aspects of the testing process that was described. Specifically:

Table 21 SHIELD's integration and testing plan in counterpoint with the testing process suggested by Telefónica.

Documentation	Documentation is not addressed within D5.1 but D3.1/D3.2 contain the specifications and design of the SHIELD vNSFs. The SHIELD wiki is also used for preliminary documentation. The bitbucket and github repositories of the project will feature the official documentation information.
Resource requirements	Network scaling is covered under requirement PF07 and D5.1 defines scale-in and scale-out platform tests for the vNSFs (coded as PLT in D5.1/Annex B). On-the-fly autoscaling is not considered at the moment since it is not supported in OSM. Specific resource requirements for the SHIELD vNSFs are included in Section 3 as they currently stand; further optimisation leading to better resource allocation can be envisioned. Y1 vNSFs were deployed in 20-40GB(storage)/4GB(memory) flavours. The number of VNFC in a specific vNSF and the need for internal storage directly impacts the size of the storage required. CPU resources are dependent on the functionality of the vNSF and can range from a single virtual processor to multiple ones. This document also includes resource requirements for all components in its Requirements mapping subsections.
Instantiation	Instantiation and management of the vNSF lifecycle is covered in PF01-PF02 and multiple platform tests are defined in D5.1 to test instantiation, policy configuration, scale-in and scale-out etc. Communication with the DARE is covered in the test plan and it involves the vNSFs sending monitoring information to the DARE and receiving remediation actions through OSM.
Management	
Non-Functional testing	The non-functional testing of vNSFs' performance is assessed in the Performance and Usability tests (coded as PUT in D5.1/Annex B) as well as some platform tests regarding scalability with respect to data volume (up to near-operational telco conditions). vNSF hardening is addressed and it effect on vNSF performance is also taken into account. Various performance metrics are defined. Availability defines total uptime until a failure as a KPI that is closely related to the Durability requirement, although durability tests can be added, i.e. continuous running for periods of 4hs, 8hs, 24hs, 48hs should be performed for Y1-Y2 vNSFs to assess their performance. Failure recovery should be sufficiently addressed; vNSF behaviour under platform and VM failures/restarts etc. should be assessed and there is no incident reporting from the vNSFs at the moment. This work can be roadmapped for future vNSF developments.
Functional testing	Functional Testing for the SHIELD vNSFs is already covered in the Integration plan (D5.1 Section 3). This regards the use of cybersecurity tools, specialised operating systems or frameworks to generate traffic and cyberattack patterns to test SHIELD cybersecurity Network Services, DARE detection and remediation as well as individual vNSFs. These are defined as Service Tests in D5.1 (coded as SET in D5.1/Annex B). Platform tests that regard vNSF interactions with the vNSFO and the DARE are also covered. Testing of vNSF performance include different protocols although different packet sizes (outside of UDP/TCP/ICMP protocols) can be tested in Y2 as well. Latency and packet loss are considered although jitter is

	considered in fewer tests and can be more explored. The performance of vNSFs is assessed in the Performance and Usability tests (coded as PUT in D5.1/Annex B).

Based on the D5.1 plan and during Y1, SHIELD performed integration and validation tests on the entire SHIELD platform including the vNSF ecosystem. The results were demonstrated in 3 scenarios, specifically:

Demo 1: Detection of data exfiltration through DNS tunnelling,

Demo 2: On-boarding a SHIELD vNSF and mitigation of DDoS attacks, and

Demo 3: NFVI/vNSF attestation.

The SHIELD Y1 demos were presented internally during the project's 4th General Assembly and the project's first Review Meeting. They were also presented publicly during:

- **the ENISA Bonding EU Cyber Threat Intelligence (CTI – EU) workshop** that took place in October 30th -31st 2017 in Rome, Italy.
- **the IEEE Conference on Network Function Virtualization and Software Defined Networks (IEEE NFV/SDN)**, held November 6-8, 2017 in Berlin, Germany, where they received the Best Demo Award among 22 competitors. The demo was submitted under the title "NFV-based network protection: the SHIELD approach".

The videos presenting the demos are publicly available in the project's YouTube channel[3]. Specifically, demos 2 and 3 are mostly focused on the vNSF ecosystem. Demo 2 showed the on-boarding process for a vNSF and demonstrated end-to-end functionality, including communication of the vNSF with the DARE Security Analytics module (TALAIA) that detects the attack, the recommendation and remediation engine that provides the mitigation actions (POLITO), the dashboard where the user visualizes the recommendation and selects whether to apply it, and the vNSF receiving the mitigation action through the vNSFO. Demo 3 focused on the Trust Monitor and showed the attestation process for the NFVI and a SHIELD vNSF. D5.1 provides a more comprehensive presentation of the demo results [13]. Deliverable D2.2 includes the basic requirements for the upcoming SHIELD Demonstrations, while D5.2 will focus on presenting results from all the SHIELD demo scenarios. The next section defines the performance and availability requirements for carrier-grade vNSFs that would be the final targets for testing and validation.

5.3. Requirements for carrier-grade performance

Carrier-grade performance relates to the "**five-nines"**, which translates to **99,999% uptime** and **"extremely high availability".** Moreover, customers only care that the service they are paying for and rely on works as offered. In SHIELD there are two factors that affect the customers' service:

- The SHIELD vNSFs, and
- the DARE.

[3] https://www.shield-h2020.eu/about/social-networks.html

SHIELD features vNSFs that monitor the traffic without intrusion in the communications by working with a traffic copy (e.g. an IDS) as well as vNSFs that they are inserted in the flow of the traffic (e.g. a firewall). A fail in the first category may affect the service but not break the customer communication. A fail in the second category could be more critical. For this reason, the degree of reliability and protection have to be greater for vNSFs that are inserted in the flow of the traffic.

Regarding the DARE, a failing in such system can affect the service of a great number of users but not affect their communications. Since SHIELD focuses mainly on offering a security service, it is highly critical to its business case to have the maximum availability of this module.

Availability is only one part of the concept of carrier-grade performance. Several other considerations must be taken into account: Reliability/Dependability, Maintainability, Manageability, Scalability, Accountability, and Durability, all collected as requirements in the D2.2.

With these considerations, it is important to validate the KPIs collected in the D2.2 using procedures defined in D5.1. The analysis of these tests has to give as result in **a loss of service lower than 6 minutes in the worst case**. For critical vNSFs where is necessary to deploy high availability to avoid interrupt the customers communications it is necessary to test this high availability with the vNSF. Regarding DARE it is important to verify redundancy with one node failure. Measure recovery time in reasonable time (e.g. less than 5 minutes). Furthermore, the selected hardware will need to meet the carrier grade specification and fulfil the requirements of MTBF (Mean-Time-Between-Failures) and MTTR (Mean-Time-To-Repair):

(Equation 1) $$Availability = \frac{MTBF*100}{MTBF+MTTR} = 99{,}999\%.$$

5.4. Evolution of a service marketplace

SHIELD's Store component focuses on providing a secure and trustable virtual network security function and network service software catalogue. It validates all the on-boarded network solutions even before attempting the deploying of each one of them in the NFVI environment by SHIELD's Orchestrator. The Store also accommodates information on what personal data is collected by the vNSFs composing a service and the rationale for its collection. Hence, the Store is in a place to provide a **high quality standard repository of network functions** already addressing privacy concerns such as GDPR compliance.

In order to evolve the Store from a repository to a service marketplace, some features such as **user management** and **billing mechanisms** need to be developed. In SHIELD, these are provided by the Dashboard [10]. Following this rationale, SHIELD's marketplace concept is based on both the Store and the Dashboard. This architectural decision allows a clear separation of the main-feature plug-n-play blocks of the marketplace (user/role management, billing mechanism, function and service secure repository). This choice enables the easy creation of network services that accommodate different business-oriented needs as well as the individual evolution of each one of them.

The proposed marketplace is envisioned to be able to serve both a single instance of a SHIELD deployment (to be demonstrated in the context of SHIELD project) as well as multiple SHIELD deployments or other solutions compatible with SHIELD package format. The second scenario is the most profitable scenario since it would **provide a centralised place for service/function providers to share their solutions among the different potential clients**. This scenario holds the highest value for the creators and maintainers of such a marketplace that may receive, for instance, a fee of the transactions performed, but also for the all the envisioned actors of such an ecosystem. Focusing on the function/service providers, a centralised marketplace would allow using **a single market to provide their solution portfolio** to the current and potential clients. Maintenance and software upgrade routines are eased by this centralised marketplace environment. This approach would in fact allow the creation of a vNSF/NS ecosystem thus **promoting the competition** (both in terms of quality as well as pricing) between providers of similar/alternative solutions. The benefit arising from this would impact the service providers offering services to end users, either by allowing to use software in this marketplace to provide services as well as to use this marketplace also to provide services to end users. Internet service providers could therefore use this marketplace not only to retrieve the building blocks (virtual network functions) to be used in their services (either developed in-house or ultimately being also developed and exposed in this marketplace) but also to advertise to potential end users the services provided and that they can acquire either for their residential or commercial installations. By this means, this marketplace can not only boost the vNSF and NS development ecosystem but also provide **visibility to potential end users** on the services provided by different internet service providers thus allowing comparing the added value of each internet service provider as well as the prices associated with each one.

The marketplace is intended to not only allow different providers (both business-to-business and business-to-consumer) to share their solution to potential and effective clients but also managing all the monetisation associated to this ecosystem. Billing model is the component associated with this set of features, thus responsible for managing the money flow and authorizations associated with both the acquisition of vNSFs or NSs from Internet Service Providers (B2B) as well as the acquisition of NSs from end users (B2C). Different billing strategies should be accommodated in this billing framework allowing consumers to adapt to what best suits its internet habits or business model. The charging rationale for B2B and B2C should be decoupled allowing the billing to be as adaptable to the market needs as possible. Some of the currently envisioned strategies for the B2C market are listed below:

- **Charge by instantiation of a service or function:** The client pays a rent for the time each service/function is running and dedicated to him.
- **Charge by resource consumption of the services/functions** instantiated in the context of a client: The client pays a rent on the hardware resources used to instantiate the services/functions running in the context of his account.
- **Charge by added value service bundle**: The client pays a fixed rate associated with a set of added value features, i.e. different security enhancements being provided on the context of its connections regardless of what services or functions are currently instantiated to achieve them.

Now focusing on the B2B market, listed below are some potential billing methodologies that should be provided in the context of the marketplace:

- **Fixed charge value by the number of instantiations** of a given service or function. Software providers would charge a fixed value associated with the number of times a given function/service from its authority is instantiated by a service provider.

- **A percentage on the profit that internet service provider bills on end clients** by using one of the services/functions developed by a software provider.
- **Licensing payment effectively acquiring a given function/service payed at acquisition time** and not depending on when and how many times a service/functions is instantiated.

The billing framework requires **usage statistics** (i.e. when a given function is instantiated, total uptime, etc) in order to monetise the offered services. Hence, the billing framework should provide an interface allowing the Orchestrator and/or OpenStack to share instantiation information at the NFVI level. Resource usage statistics can also be collected through OpenStack Ceilometer. A full implementation of the billing mechanisms, however, is out of the scope of the project. This would require the additional implementation of **accounting mechanisms, rights management, service level agreement assurance,** etc. Further work on definition of **clear liability borders** in an ecosystem of multiple stakeholders is also required. Certification, as discussed in Sections 4 and 5, would also increase trust in the service market and contribute to its adoption.

Additional business models could be added to this component allowing for instance the **appearance of auctions** enabling both sellers and consumers to bid on the providing/acquisition of software artefacts managed by the marketplace. Furthermore, the marketplace's vision allows also for instance **a penalisation billing rationale** allowing internet service providers to protect themselves if client's SLAs are breached due to the malfunction of third party software. To achieve so, besides receiving monitoring usage information, the billing framework should also receive error reports to apply fines to the entities responsible for the development of the defective functions/services. These fines could be billed for instance following the following principles:

- **Charged rate by detected anomaly based on a severity level** (Very low, Low, Medium, High, Very High, Critical)
- **Charged a percentage rate on the profit** this entity would make for the usage of the defective function/service
- **Charged a percentage level on the fine applied to internet service provider** for not fulfilling the SLAs of the client

Aligned with the previous penalisation philosophy, the marketplace could also provide **a reputation-based ranking system** associated with the fulfilment or failure of the provided software comparing with the promised and charged behaviour. This would enable buyers to avoid providers/services/functions with a high number of malfunctions.

User management is another important feature of a marketplace allowing different users with different roles/scopes to interact with the system with different goals. These features and roles are envisioned and implemented in the scope of Dashboard following the envisioned actors for SHIELD as a platform and not dedicated solely to the requirements of the marketplace. However, if the marketplace is to be instantiated in a way to serve multiple SHIELD deployments (or other solutions compatible with SHIELD's envisioned marketplace) user management implementation would have to be instantiated in the scope of the marketplace allowing this solution to have an intended and trustable user management system. A minor adaptation would have to be performed, trimming some roles that are not valid actors in the context of the marketplace but need to exist in the scope of SHIELD solution.

Both vertical and horizontal business scenarios can be supported in this context. Currently, the marketplace already provides privacy policy features (through Store's GDPR features) as well as pre-validation features ensuring that the onboarded software is compatible with the Orchestration environment (e.g. Store's vNSF syntax, integrity and topology validation). Higher quality and sustainability of the marketplace service ecosystem, would require existing certification marks to be integrated in the marketplace's ecosystem. Providing a SHIELD certification that would ensure to potential bidders/buyers that certain functions or services have a high-quality standard could be envisioned although it would require a rigorous testing process and the existence of a SHIELD accredited association (as per ISO 17065). Even though this is not envisioned in the current version of the marketplace, this implementation is possible by first roadmapping an analysis of the roles and user profiles in this ecosystem as well as the certification acquisition, administration and eviction processes for each function or service.

6. Conclusions

6.1. Status of vNSF ecosystem

This document presents the finalised technical details of the vNSF ecosystem, starting with the high-level architecture and design to the specifications and implementation choices. Section 2 deals with the high-level picture of the SHIELD vNSF platform, its purpose and interconnections between components, whilst the latter presents low-level details, such as the specifications to cover, its mapping with the requirements defined in D2.1 and the decisions regarding implementation aspects.

As depicted in the document, SHIELD's vNSF ecosystem is composed by the vNSFs, Store, Orchestrator and Trust Monitor components. High-level architecture is provided per each of these components, taking into consideration the requirement specification as well as SHIELD's use cases. On the other hand, the high-level specifications, especially for the vNSFs and vNSFO, have been defined by following the recommendations and specifications of ETSI, considering it as the main standardisation body in the area. This alignment is one of the main goals of the consortium since it greatly promotes and eases the dissemination and exploitation of SHIELD's results into this standardisation body or other reference ecosystems. An example of the envisioned collaboration deals with the contribution of extensions developed within the project into some of the current standardisation bodies.

The low-level details specified for each of the scoped components result in an important asset for the next development phase of the project, concerned with the implementation of such components. Specifically, the definition of the intra and inter-connectivity workflows makes it easier to agree on the responsibilities and behaviour of each component, how these will be implemented, and which features will be provided by each one of them ensuring its integration at a later stage of the project. The specifications of these connections (provided in Section 3) took into consideration the full set of components involved in the architecture, including some from the analytics and visualisation part which is comprehensively covered in D4.2.

The details on the implementation per component indicate the intention to reuse the results of previous projects and other open-source solutions as much a possible; covering a fair amount of functionality and thus allowing to better focus on innovative aspects not yet covered by the community. As a result, SHIELD enters a new development cycle in Y2, where new vNSFs will be developed, with novel functionalities derived from the updated requirements in D2.2.

The D2.2 deliverable introduced new ethical and regulatory compliance requirements for the SHIELD platform. Section 4 also addressed these new requirements and provided a specifications template for regulatory compliance with the EU legal landscape. It further discussed how these specifications can be introduced in context of the Store, analysed some best practices for GDPR compliance and touched upon GDPR certification as a future development.

Finally, this document also discussed how to improve the exploitation potential of SHIELD vNSFs by means of appropriate certification and qualification (Section 5). Certification environments driven by mobile suppliers (like Nokia or Huawei) are available although they cannot be used in the context of the SHIELD Store due to major differences in deployment configurations. An

OSM-based certification could easily be exploited in the current deployment of the SHIELD architecture. Some basic requirements on the tests required by a certification process were also included, with input from Telefónica, which is a major telco operator. Compliance, certification and the appropriate billing mechanisms are key aspects towards the evolution of the project Store in a future service marketplace.

6.2. Future work

The work of T3.1 "vNSF infrastructure and software specifications, design and architecture" concludes with this document that provides the results of the second design cycle in SHIELD. Tasks 3.2-3.5 continue the work with the development of all SHIELD vNSFs, the vNSF Store, and the attestation framework. A rigorous testing plan is in place to validate the new components, based on the work in D5.1. Results of WP3 activities will be presented in SHIELD's upcoming demonstrations as presented in the D2.2 demonstration roadmap. D3.3 "Integrated secure framework ready for experiments" will report all the results stemming from WP3 developments and accompany the delivery of the WP3 prototypes.

REFERENCES

[1] ETSI, "Network Functions Virtualisation (NFV); Virtual Network Functions Architecture".

[2] SONATA Consortium, "The Sonata Project," [Online]. Available: http://sonata-nfv.eu. [Accessed March 2018].

[3] ETSI OSM, "Open Source MANO," [Online]. Available: https://osm.etsi.org/ . [Accessed March 2018].

[4] The SECURED consortium, "The SECURED project," [Online]. Available: http://www.secured-fp7.eu. [Accessed March 2018].

[5] "Open Attestation project," [Online]. Available: https://github.com/OpenAttestation/OpenAttestation/tree/v1.7. [Accessed March 2018].

[6] "Apache Cassandra 2 project," [Online]. Available: https://cassandra.apache.org/ . [Accessed March 2018].

[7] L. Jacquin, A. L. Shaw and C. Dalton, "Towards trusted software-defined networks using a hardware-based Integrity Measurement Architecture," in *2015 1st IEEE Conference on Network Softwarization (NetSoft)*, London, UK, 2015.

[8] The SHIELD consortium, "D2.1 Requirements, KPIs, design and architecture," 2017.

[9] The SHIELD consortium, "D2.2 Updated requirements, KPIs, design and architecture," 2018.

[10] The SHIELD Consortium, "D4.1 Specifications, design and architecture for the usable information-driven engine," 2017.

[11] SHIELD Consortium, "Updated specifications, design and architecture for the usable information-driven engine," 2018.

[12] The SHIELD Consortium, "D3.1 Specifications, design and architecture for the vNSF ecosystem," 2017.

[13] The SHIELD Consortium, "D5.1 Integration results of SHIELD HW/SW modules," 2018.

[14] "mcTLS," [Online]. Available: - http://mctls.org/ . [Accessed March 2018].

[15] "ACME STAR protocol," [Online]. Available: https://tools.ietf.org/html/draft-ietf-acme-star-03. [Accessed March 2018].

[16] POLITO, "TCP STatistics and Analysis Tool," [Online]. Available: http://tstat.polito.it. [Accessed March 2018].

[17] "RFC 8329," [Online]. Available: https://datatracker.ietf.org/doc/rfc8329. [Accessed March 2018].

[18] "Trusted Computing," [Online]. Available: https://trustedcomputinggroup.org/trusted-computing/ . [Accessed March 2018].

[19] CHARISMA consortium, “Charisma D3.2 deliverable,” [Online]. Available: http://www.charisma5g.eu/wp-content/uploads/2015/08/CHARISMA-D3.2_v1.0.pdf. [Accessed March 2018].

[20] “Snort IPS,” [Online]. Available: https://www.snort.org/ . [Accessed March 2018].

[21] “Barnyard2 a dedicated spooler for snort's unified2 binary output format,” [Online]. Available: https://github.com/firnsy/barnyard2. [Accessed March 2018].

[22] “PulledPork fo Snort and Suricata rule management (from Google Code),” [Online]. Available: https://github.com/shirkdog/pulledpork. [Accessed March 2018].

[23] “Ruby on Rails Application for Network Security Monitoring,” [Online]. Available: https://github.com/Snorby/snorby. [Accessed March 2018].

[24] “ntop nDPI,” [Online]. Available: http://www.ntop.org/products/deep-packet-inspection/ndpi/ . [Accessed March 2018].

[25] “OpenDPI code repository,” [Online]. Available: http://code.google.com/p/opendpi/ . [Accessed March 2018].

[26] “PF_RING,” [Online]. Available: http://www.ntop.org/products/packet-capture/pf_ring/. [Accessed March 2018].

[27] “Data Plane Development Kit,” [Online]. Available: http://dpdk.org/ . [Accessed March 2018].

[28] “Structured Threat Information Expression (STIX 2.0) CTI documentation,” [Online]. Available: https://oasis-open.github.io/cti-documentation/. [Accessed March 2018].

[29] “mitmproxy,” [Online]. Available: https://mitmproxy.org/. [Accessed march 2018].

[30] “CertBot,” [Online]. Available: https://certbot.eff.org/. [Accessed March 2018].

[31] “Boulder GitHub page,” [Online]. Available: https://github.com/letsencrypt/boulder. [Accessed March 2018].

[32] “Let's encrypt,” [Online]. Available: https://letsencrypt.org/. [Accessed March 2018].

[33] “Nfdump,” [Online]. Available: http://nfdump.sourceforge.net/. [Accessed March 2018].

[34] “The iptables project,” [Online]. Available: https://www.netfilter.org/projects/iptables/index.html. [Accessed March 2018].

[35] “OpenBSD PF - User's Guide,” [Online]. Available: https://www.openbsd.org/faq/pf/. [Accessed March 2018].

[36] “ModSecurity,” [Online]. Available: https://www.modsecurity.org/. [Accessed March 2018].

[37] “HTTPD Apache Hypertext Transfer Protocol Server,” [Online]. Available: https://httpd.apache.org/docs/2.4/programs/httpd.html. [Accessed March 2018].

[38] “OWASP ModSecurity CRS,” [Online]. Available: https://github.com/SpiderLabs/owasp-modsecurity-crs. [Accessed March 2018].

[39] “Apache Traffic Server,” [Online]. Available: http://trafficserver.apache.org/. [Accessed March 2018].

[40] "HAProxy - The reliable, high-performance TCP/HTTP load balancer," [Online]. Available: http://www.haproxy.org/. [Accessed March 2018].

[41] "Varnish HTTP Cache," [Online]. Available: https://varnish-cache.org/. [Accessed March 2018].

[42] "nginx - Flawless application delivery for the modern web," [Online]. Available: https://www.nginx.com/. [Accessed March 2018].

[43] "NAXSI is an open source, high performance, low rules maintenance WAF for NGINX," [Online]. Available: https://github.com/nbs-system/naxsi. [Accessed March 2018].

[44] "Eve. The Simple way to REST," [Online]. Available: http://python-eve.org/. [Accessed March 2018].

[45] "Flask-RESTful," [Online]. Available: https://flask-restful.readthedocs.io/. [Accessed March 2018].

[46] "Cerberus for a lightweight and extensible data validation," [Online]. Available: http://docs.python-cerberus.org/. [Accessed March 2018].

[47] "MongoDB for GIANT ideas," [Online]. Available: https://www.mongodb.com/. [Accessed March 2018].

[48] "Behavior Driven Development," [Online]. Available: https://www.agilealliance.org/glossary/bdd/. [Accessed March 2018].

[49] "TeNOR Repository," [Online]. Available: https://github.com/T-NOVA/TeNOR . [Accessed March 2018].

[50] "OpenBaton," [Online]. Available: https://openbaton.github.io . [Accessed March 2018].

[51] "Open Source MANO releas 2," [Online]. Available: https://osm.etsi.org/wikipub/index.php/OSM_Release_TWO . [Accessed March 2018].

[52] "Open Source MANO release 2 - Whitepaper," [Online]. Available: https://osm.etsi.org/images/OSM-Whitepaper-TechContent-ReleaseTWO-FINAL.pdf . [Accessed March 2018].

[53] "OpenCIT project," [Online]. Available: https://github.com/opencit/opencit/wiki/Open-CIT-2.2-Product-Guide. [Accessed March 2018].

[54] "Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Da," [Online]. Available: http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=celex%3A32016R0679. [Accessed March 2018].

[55] "Regulation (EU) 2015/2120 of the European Parliament and of the Council of 25 November 2015 laying down measures concerning open internet access and amending Directive 2002/22/EC on universal service and users' rights relating to electronic communications," [Online]. Available: http://eur-lex.europa.eu/legal-content/EN/TXT/?qid=1520936363145&uri=CELEX:32015R2120. [Accessed March 2018].

[56] "Directive 2002/58/EC of the European Parliament and of the Council of 12 July 2002 concerning the processing of personal data and the protection of privacy in the electronic communications sector (Directive on privacy and electronic communications)," [Online]. Available: http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=celex%3A32002L0058. [Accessed March 2018].

[57] "Proposal for a REGULATION OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL concerning the respect for private life and the protection of personal data in electronic communications and repealing Directive 2002/58/EC (Regulation on Privacy and Electronic Commu," [Online]. Available: http://eur-lex.europa.eu/legal-content/EN/TXT/?qid=1520936405198&uri=CELEX:52017PC0010. [Accessed March 2018].

[58] "Directive (EU) 2016/680 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data by competent authorities for the purposes of the prevention, investigation, detectio," [Online]. Available: http://eur-lex.europa.eu/legal-content/EN/TXT/?qid=1520936541517&uri=CELEX:32016L0680. [Accessed March 2018].

[59] " Directive (EU) 2016/1148 of the European Parliament and of the Council of 6 July 2016 concerning measures for a high common level of security of network and information systems across the Union," [Online]. Available: http://eur-lex.europa.eu/legal-content/EN/TXT/?qid=1520936601568&uri=CELEX:32016L1148. [Accessed March 2018].

[60] "Council Directive 2000/78/EC of 27 November 2000 establishing a general framework for equal treatment in employment and occupation," [Online]. Available: http://eur-lex.europa.eu/legal-content/EN/TXT/?qid=1520936644204&uri=CELEX:32000L0078. [Accessed March 2018].

[61] "Charter of Fundamental Rights of the European Union (2000/C 364/01)," [Online]. Available: http://www.europarl.europa.eu/charter/pdf/text_en.pdf. [Accessed March 2018].

[62] "Treaty of Amsterdam amending the Treaty on European Union, the Treaties establishing the European Communities and certain related acts," [Online]. Available: http://www.europarl.europa.eu/topics/treaty/pdf/amst-en.pdf. [Accessed March 2018].

[63] "Treaty of Lisbon amending the Treaty on European Union and the Treaty establishing the European Community, signed at Lisbon, 13 December 2007," [Online]. Available: http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=celex%3A12007L%2FTXT.

[64] "Council of Europe Recommendation CM/Rec(2010)13 and explanatory memorandum," [Online]. Available: https://rm.coe.int/16807096c3. [Accessed March 2018].

[65] "European Agency of Fundamental Rights (FRA) Handbook on non-discrimination case law," [Online]. Available: http://fra.europa.eu/en/publication/2011/handbook-european-non-discrimination-law-2011-edition. [Accessed March 2018].

[66] K. E. Himma and H. T. Tavani, The Handbook of Information and Computer Ethics, Hoboken, New Jersy: Wiley, 2008.

[67] A. Cavoukian, Privacy by Design - the 7 foundational principles, www.privacybydesign.ca, 2011 (rev).

[68] "Data protection impact assessment," Information Commissioner's Office, UK , [Online]. Available: https://ico.org.uk/for-organisations/guide-to-the-general-data-protection-regulation-gdpr/accountability-and-governance/data-protection-impact-assessments/. [Accessed March 2018].

[69] "OWASP Top 10 Privacy Risks and Countermeasures," [Online]. Available: https://www.owasp.org/images/0/0a/OWASP_Top_10_Privacy_Countermeasures_v1.0.pdf. [Accessed March 2018].

[70] "PrivacyTrust," [Online]. Available: http://www.etrust.org/. [Accessed March 2018].

[71] “HITRUST Alliance,” [Online]. Available: http://www.hitrustalliance.net/. [Accessed March 2018].

[72] “Health Insurance Portability and Accountability Act,” 2016. [Online]. Available: http://www.legalarchiver.org/hipaa.htm. [Accessed March 2018].

[73] “ISO/IEC 27001 Information Security Management Systems,” [Online]. Available: https://www.iso.org/isoiec-27001-information-security.html. [Accessed March 2018].

[74] “Information technology -- Security techniques -- Code of practice for protection of personally identifiable information (PII) in public clouds acting as PII processors,” [Online]. Available: https://www.iso.org/standard/61498.htm. [Accessed March 2018].

[75] “ISO/IEC 29100:2011,” [Online]. Available: https://www.iso.org/standard/45123.html. [Accessed March 2018].

[76] “OASIS PMRM,” [Online]. Available: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=pmrm. [Accessed March 2018].

[77] “Nokia CloudBand,” [Online]. Available: https://networks.nokia.com/solutions/cloudband-ecosystem . [Accessed March 2018].

[78] “Huawei NFV Integration Service,” [Online]. Available: http://www.huawei.com/en/news/2017/11/NFV-Integration-Service-Innovation-Award . [Accessed March 2018].

[79] Red, Hat, “Momentum has grown for VNF certification,” [Online]. Available: https://www.redhat.com/en/blog/momentum-has-grown-vnf-certification. [Accessed March 2018].

[80] “Telefónica NFV Reference Lab,” [Online]. Available: http://www.tid.es/long-term-innovation/network-innovation/telefonica-nfv-reference-lab . [Accessed March 2018].

[81] “Automated Certificate Mangement Environment (ACME),” [Online]. Available: https://datatracker.ietf.org/wg/acme/charter/ . [Accessed March 2018].

[82] “mcTLS code repository,” [Online]. Available: https://github.com/scoky/mctls . [Accessed March 2018].

[83] “Linux Integrity Measurement Architecture,” [Online]. Available: https://sourceforge.net/p/linux-ima/wiki/Home/ . [Accessed March 2018].

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

Acronym	Meaning
ACL	Access Control List
ACME	Automated Certificate Management Environment
API	Application Programming Interface
BSS	Business-Support System
C&C	Command and Control
CDN	Content Delivery Network
CERT	Computer Emergency Response Team
CoT	Chain of Trust
CRTM	Core Root of Trust for Measurement
CSIRT	Computer Security Incident Response Team
DARE	Data Analysis and Remediation Engine
DoS	Denial of Service
DPDK	Data Plane Development Kit
DPI	Deep Packet Inspection
EM	Element Management
EMS	EM System
FAB	Fulfilment, Assurance and Billing
FCAPS	Fault, Configuration, Accounting, Performance and Security
ETSI	European Telecommunications Standards Institute
FTP	File Transfer Protocol
GUI	Graphical User Interface
HTTP	Hyper Text Transfer Protocol
HTTPS	HTTP Secure
I2NSF	Interface to Network Security Functions
ID	Identifier

IDPS	Intrusion Detection and Prevention System
IDS	Intrusion Detection System
IMA	Integrity Measurement Architecture
IP	Internet Protocol
IPS	Intrusion Prevention System
IRC	Internet Relay Chat
ISP	Internet Service Provider
JSON	JavaScript Object Notation
LoC	Lines of Code
MAC	Media Access Control
MANO	MANagement and Orchestration
mcTLS	Multi-Context TLS
ML	Machine Learning
MSPL	Medium-level Security Policy Language
NFV	Network Function Virtualisation
NFVI	NFV Infrastructure
NS	Network Service
NSD	NS Descriptor
NSM	NS Manager
ODL	Open Day Light
ONOS	Open Network Operating System
OSI	Open Systems Interconnection
OSS	Operations Support System
PCR	Platform Configuration Registers
PF	Platform Functional
PFR	PF Requirement
PNF	Physical Network Function
PoP	Point of Presence

R&D	Research and Development
RDP	Remote Desktop Protocol
REST	REpresentational State Transfer
RO	Resource Orchestrator
SDN	Software-Defined Networking
SecaaS	Security as a Service
SMB	Server Message Block
SO	Service Orchestrator
SP	Service Provider
SPI	Stateful Packet Inspection
STAR	Short-Term and Automatically Renewed
SQL	Structured Query Language
TC	Trusted Computing
TCG	TC Group
TLS	Transport Layer Security
TM	Trust Monitor
TPM	Trusted Platform Module
TSTAT	TCP STatistic and Analysis Tool
STIX	Structured Threat Information Expression
UC	Use Case
VCA	vNSF Configuration and Abstraction
VDU	Virtual Deployment Unit
VIM	Virtual Infrastructure Manager
VL	Virtual Link
VLD	VL Descriptor
VM	Virtual Machine
VNF	Virtual Network Function
VNFC	VNF Component

vNSF	Virtual Network Security Function
VNFC	vNSF Component
vNSFD	vNSF Descriptor
vNSFFG	vNSF Forwarding Graph
vNSFFGD	vNSFFG Descriptor
vNSFM	vNSF Manager
vNSFO	vNSF Orchestrator
WAIS	Wide Area Information Server
XML	Extensible Markup Language

ANNEX A. INTRA-COMPONENT INTERACTIONS

This section provides a detailed description of the internal processes carried out within the different components, along with explanations on each step of the process.

Store

vNSF Onboarding

Onboarding a vNSF Figure 30 comprises several steps to ensure the data provided complies with the SHIELD constraints and policies. To avoid potential vNSF misbehaviour or malfunction the onboarding process encompasses an approval stage. In this stage the vNSF is registered but kept on a sandboxed state which makes it only visible to the Service Provider. Once this Service Provider deems the vNSF approved, it will be available in the Store for all the other users. Whilst the vNSF is sandboxed the Service Provider can perform any kind of validations to ensure the vNSF delivers as expected. To perform such validations a special kind of tenant may be used to provide a self-contained environment where the vNSF runs and allows the Service Provider to perform the validation in any way, shape or form, be it only the vNSF lifecycle (start/stop/etc.), any additional traffic or behaviour analysis, or operating as integrated in a NS (instantiated for the approval stage).

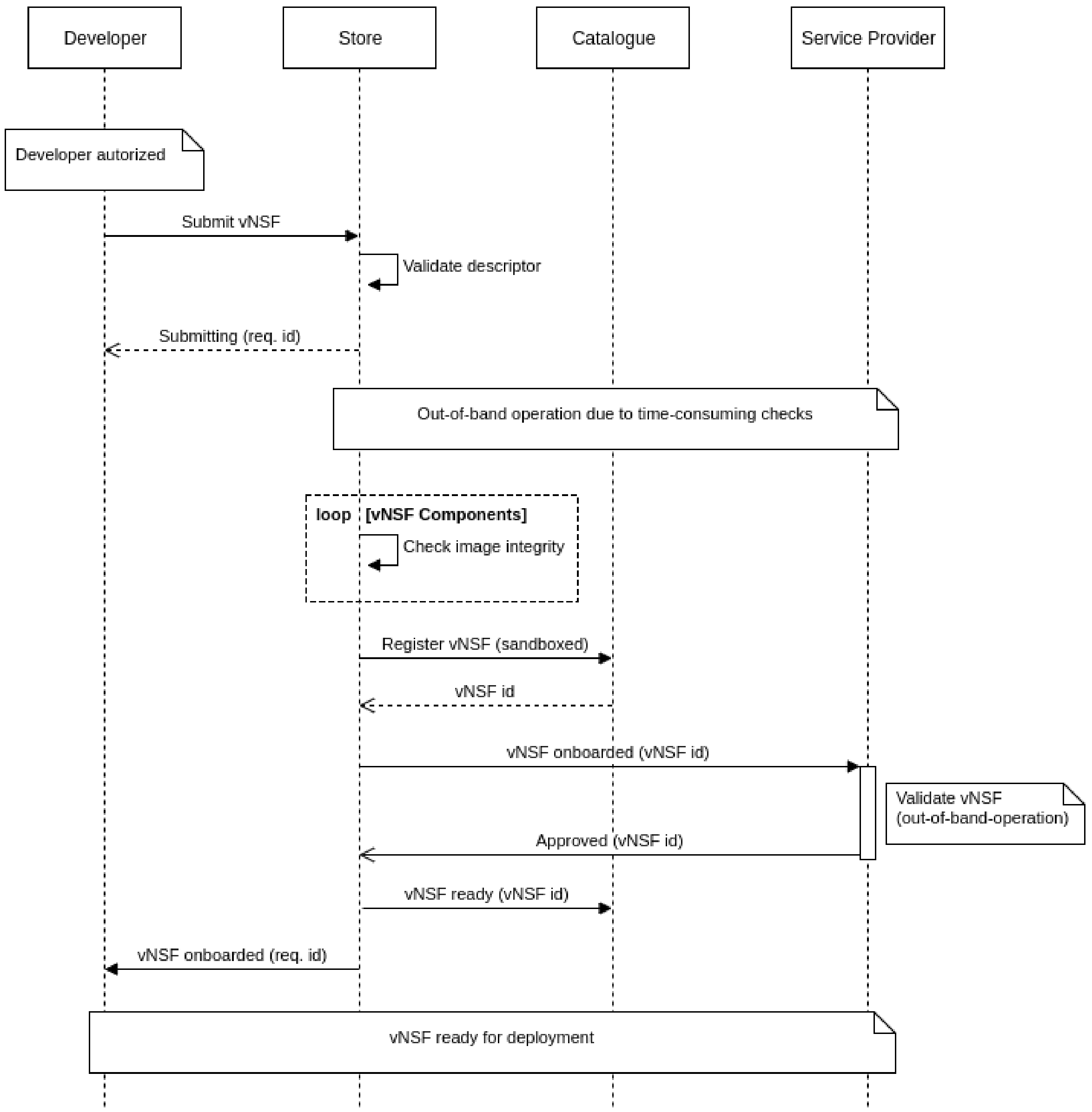


Figure 30: vNSF onboarding.

NS Onboarding: The Network Service onboarding (Figure 31) is very much like the one for vNSF with the difference being the Service Provider is the one who builds a service through chaining one or more vNSFs.

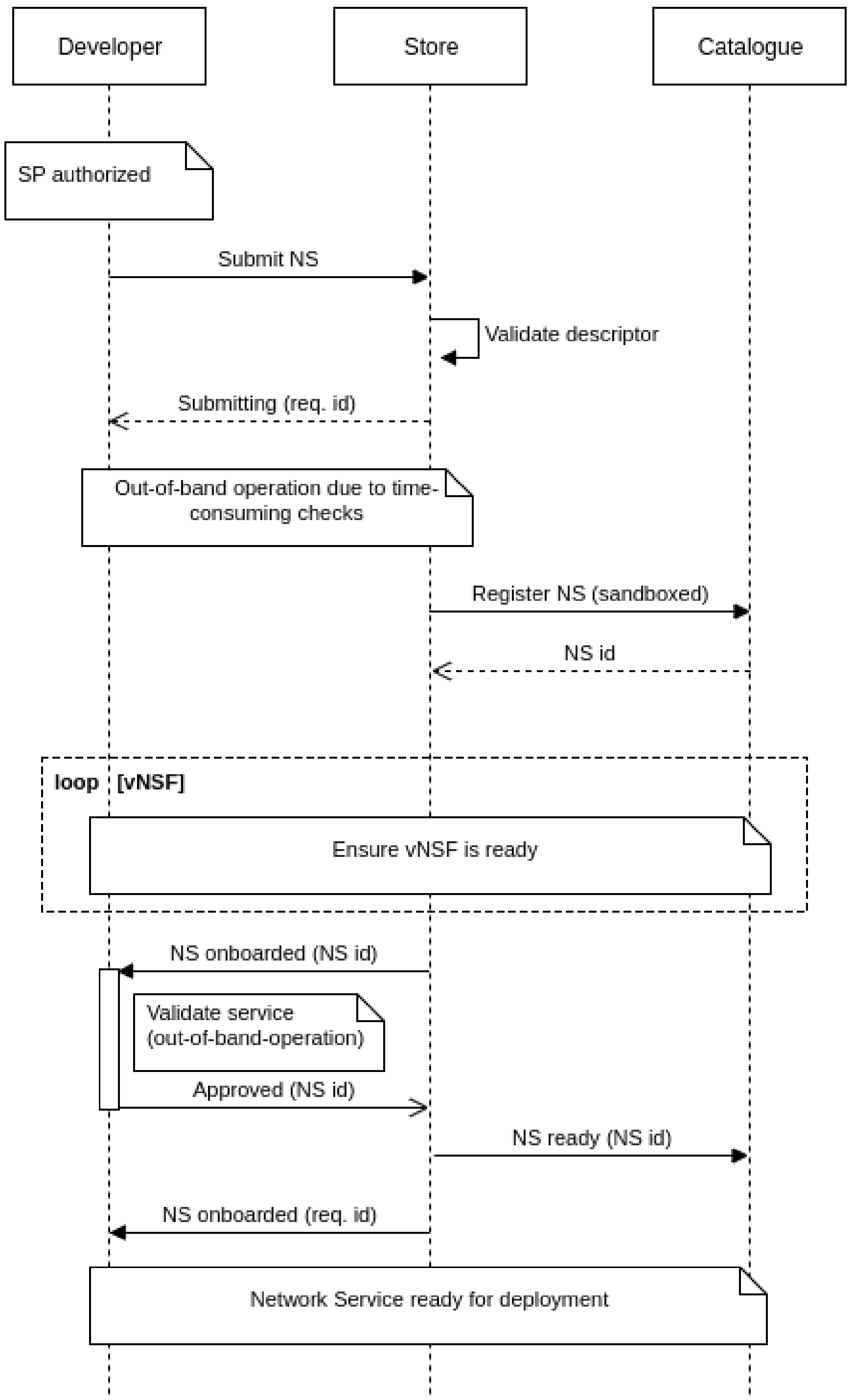


Figure 31: NS Onboarding.

vNSF/NS Onboarding Failure: The onboarding may fail (Figure 32) due to errors in the descriptors, integrity checks or final approval by the Service Provider. An example of a workflow of an onboard failure of a vNSF is provided below.

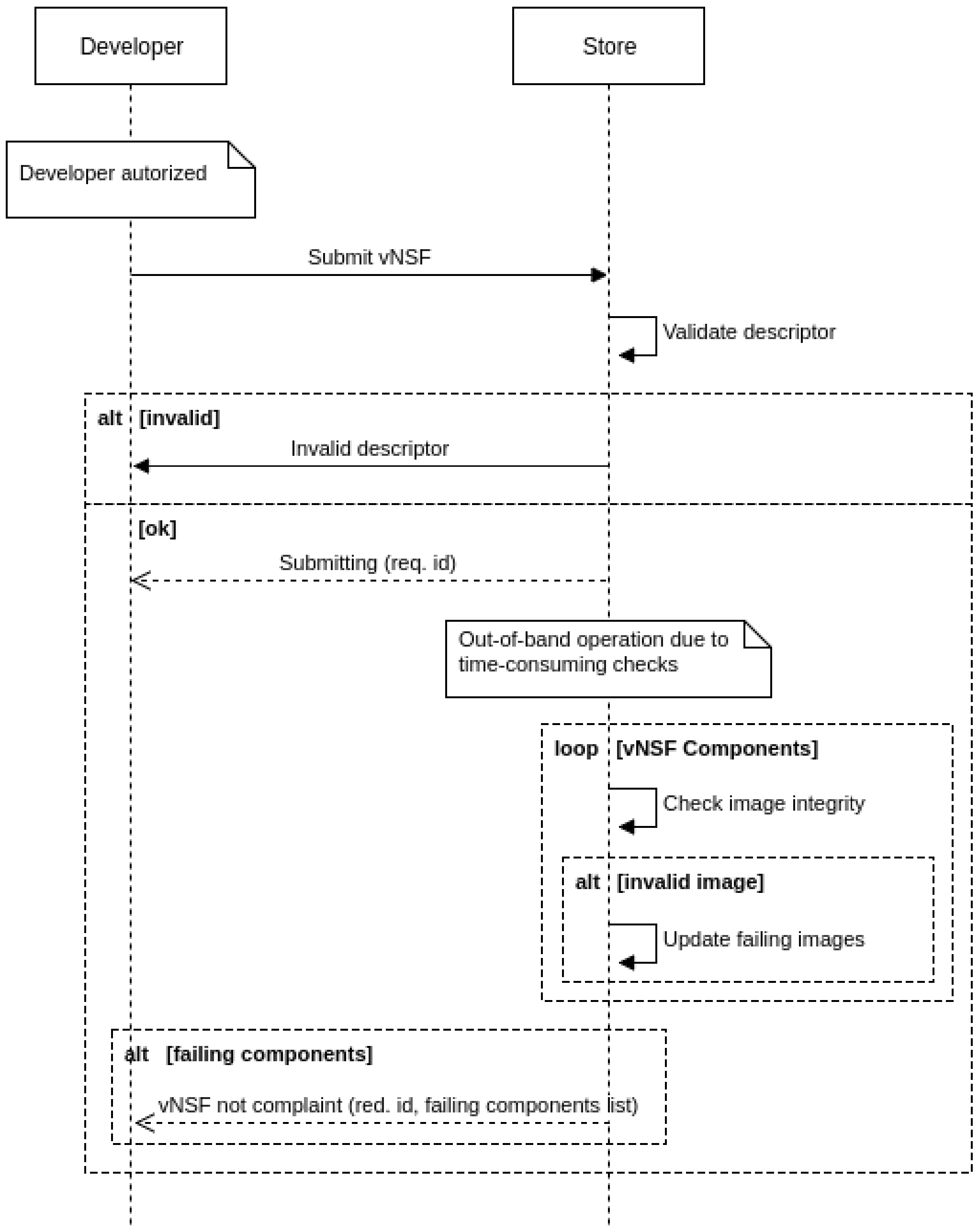


Figure 32: vNSF onboarding failure.

vNSFO Orchestrator

NS instantiation

The vNSFO exerts the instantiation workflow (Figure 33) upon deployment of a given NS, which in turn deploys the constituent vNSFs and interconnect appropriately. As part of deployment, the configuration process can occur as well in order to perform pre-boot configuration on vNSFs.

- NS deployment
    1. The vNSFO retrieves the NS descriptor from the Store.
    2. The NS descriptor is parsed to identify the constituent vNSFs and virtual links

3. The NS Manager requests the VIM on each operation, which delegates the execution to the NFVI.
4. The virtual links are defined for the vNSFs contained in the NS.
5. Upon termination of the process, the resulting status is sent to the vNSFO.

- vNSF deployment
  6. For each vNSF, the vNSFO retrieves the vNSF descriptor from the Store.
  7. The VIM downloads the image corresponding to the specific vNSF to be deployed.
  8. The request is forwarded to the NS Manager, then to the vNSF Manager.
  9. The compute nodes are allocated by the VIM and interconnected afterwards with the virtual links defined during the first stages of the NS deployment.
  10. Upon termination of the process, the resulting status is sent to the vNSFO.

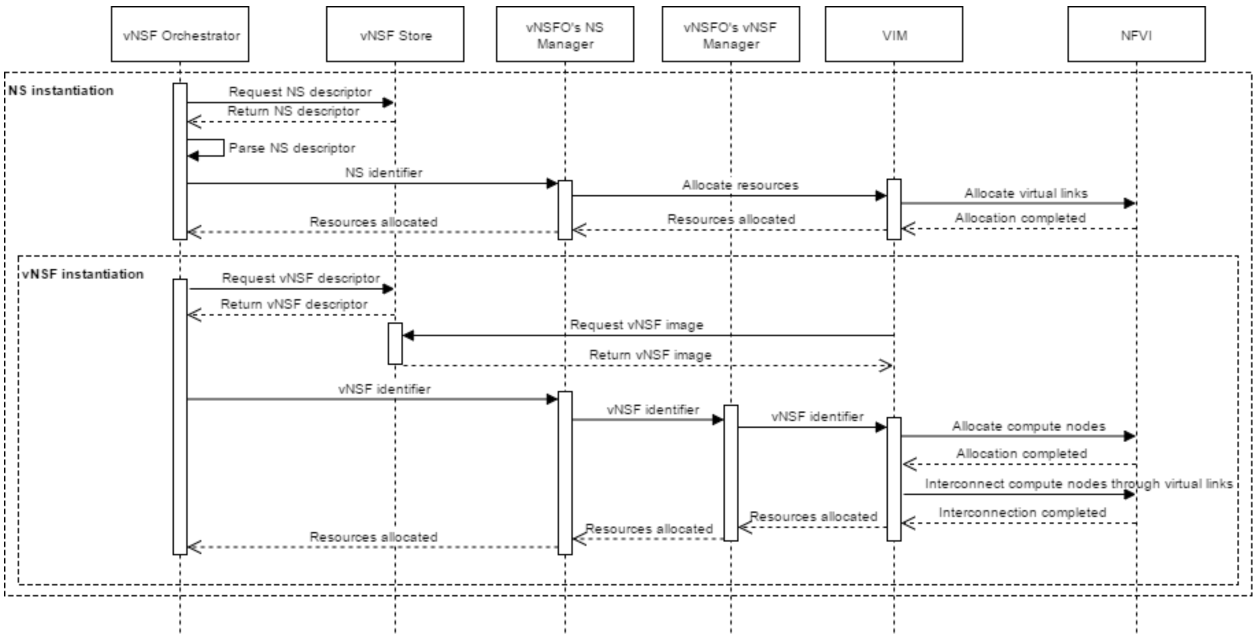


Figure 33: NS instantiation.

NS configuration: The workflow is triggered when the vNSFO receives a request for configuring a deployed NS; for instance after a user selects a recommendation from the Security Dashboard, which will provide the vNSFO with policies to apply on specific vNSFs of a given NS. Then, the vNSFO calls upon the configuration on a given vNSF (Figure 34), deploying if needed the constituent vNSFs of the service and interconnecting them.

- NS configuration
  1. The request is forwarded to the NS Manager
  2. According to the configuration requested, the NS may be required to perform a change on the virtual links interconnecting the vNSFs within the service (updating, adding or deleting them) or address configurations on vNSFs only
  3. The NS Manager requests the VIM on each operation, which delegates the execution to the NFVI
  4. Upon termination of the process, the resulting status is sent to the vNSFO
- vNSF configuration

5. For each vNSF, the request is forwarded to the NS manager, then to the vNSF Manager
6. The vNSF Manager ensures that the provided configuration policies are valid
7. If the policies are valid; the vNSF Manager makes use of specific EMs to introduce configuration into the vNSFs. The vNSFs provide endpoints to listen for configuration changes
8. Upon termination of the process, the resulting status is sent to the vNSFO.

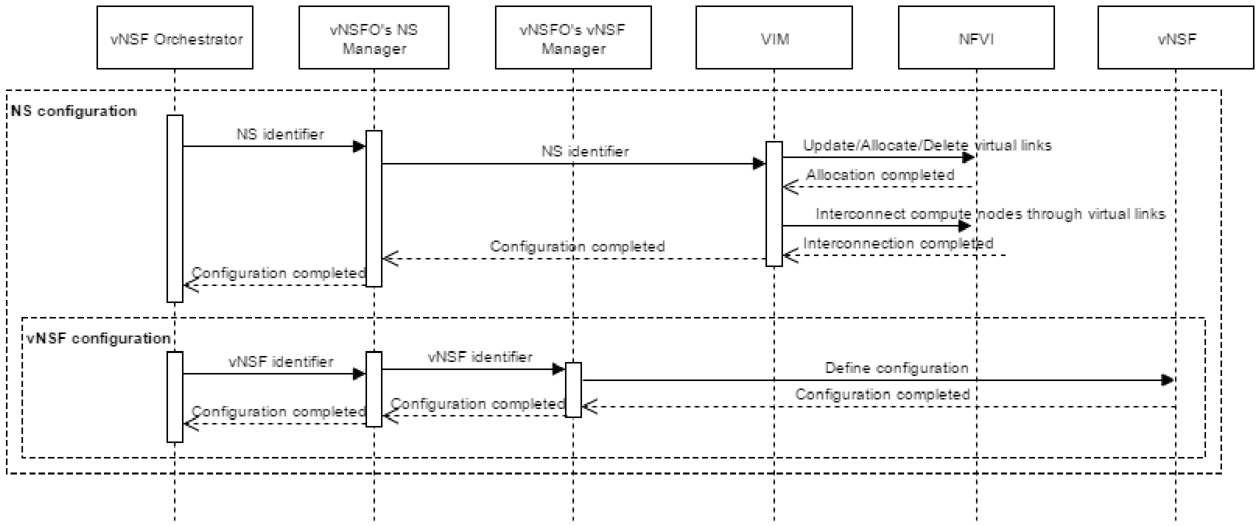


Figure 34: NS configuration.

NS monitoring: The workflow (Figure 35) is triggered when the vNSFO receives a request for monitoring a running/deployed NS.

- NS monitoring
  1. The request is forwarded to the NS Manager
  2. Using the metrics retrieved from the constituent vNSFs, the metrics are aggregated to provide information on the status of the different monitoring values. These values are described in the NSD during its registration in the Store
  3. Upon termination of the process, the resulting status is sent to the vNSFO
- vNSF monitoring
  4. For each vNSF, the request is forwarded to the NS Manager, then to the vNSF Manager
  5. The vNSF Manager asks the NFVI for metrics on the vNSF running instance (operation data on the compute nodes themselves) and requests the vNSFs for any metric on the processes running within them (such as load within specific services, etc)
  6. Upon termination of the process, the resulting status is sent to the vNSFO

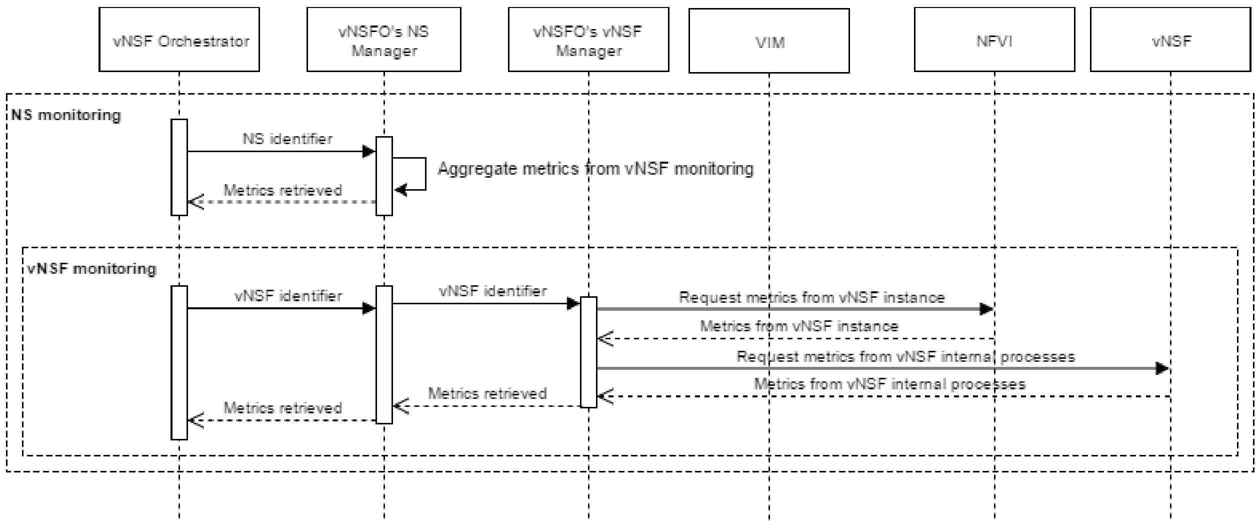


Figure 35: NS monitoring.

NS scaling: The workflow (Figure 36) is triggered when the vNSFO receives a request for scaling (reduce, increase or extend resources) an existing NS.

- NS scaling
  1. The request is forwarded to the NS Manager, then to the VIM
  2. According to the operation requested, the NS may be required to update, add or delete virtual links interconnecting the vNSFs within the service
  3. The VIM interacts with the NFVI to update the definition of the links and their interconnection with the vNSFs
  4. Upon termination of the process, the resulting status is sent to the vNSFO
- vNSF scaling
  5. For each vNSF, the request is forwarded to the NS Manager, then to the vNSF Manager
  6. The vNSF Manager forwards the request to the VIM
  7. The VIM interacts with the NFVI to remove or extend the capacity of the vNSF with additional resources
  8. Upon termination of the process, the resulting status is sent to the vNSFO.

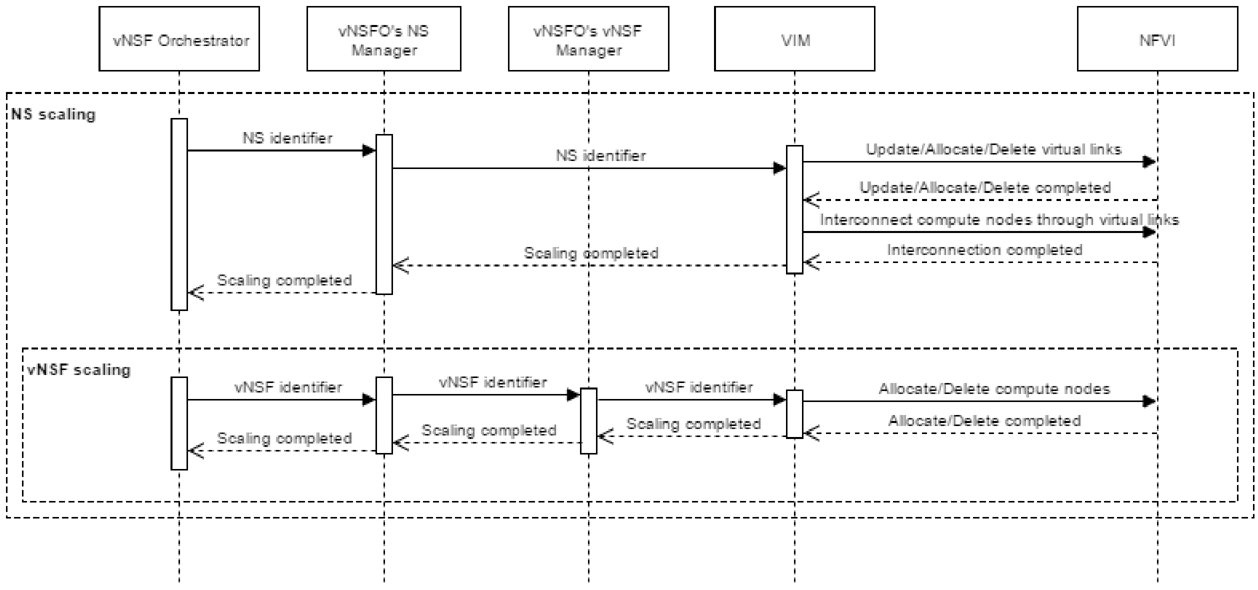


Figure 36: NS scaling.

NS termination: The workflow (Figure 37) is triggered when the vNSFO receives a request for terminating a running/deployed NS.

- NS termination:
  1. The request is forwarded to the NS Manager, then to the VIM
  2. The VIM interacts with the NFVI to remove the virtual links between the constituent vNSFs
  3. Upon termination of the process, the resulting status is sent to the vNSFO
- vNSF termination:
  4. For each vNSF, the request is forwarded to the NS Manager, then to the vNSF Manager.
  5. The vNSF Manager forwards the request to the VIM.
  6. The VIM interacts with the NFVI to terminate the vNSF and release additional physical resources associated to these.
  7. Upon termination of the process, the resulting status is sent to the vNSFO.

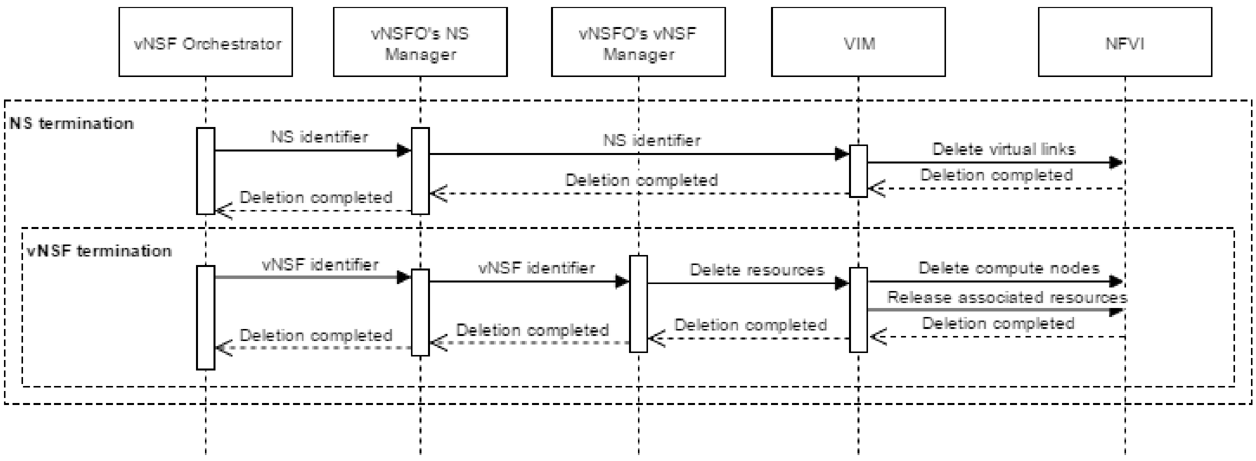


Figure 37: NS termination.

ANNEX B. INTER-COMPONENT INTERACTIONS

This section will include the description of the processes carried out between the components of the infrastructure. Each subsection will be focused on the processes initiated by a specific component.

Store

VDU Image Storage: Upon successful vNSF validation all referenced VDU images must be stored locally to allow faster instantiations. The Store provides the vNSFO with the VDU image(s) associated with the vNSF and receives a path to the image(s) storage location (Figure 38). Even though the VDU image(s) are downloaded to a Store-controlled storage location for integrity checks, these will only live in the storage controlled by the VIM. Once the images are stored by the VIM the Store do not need these anymore, so it deletes the local copy and records the final location in the Catalogue.

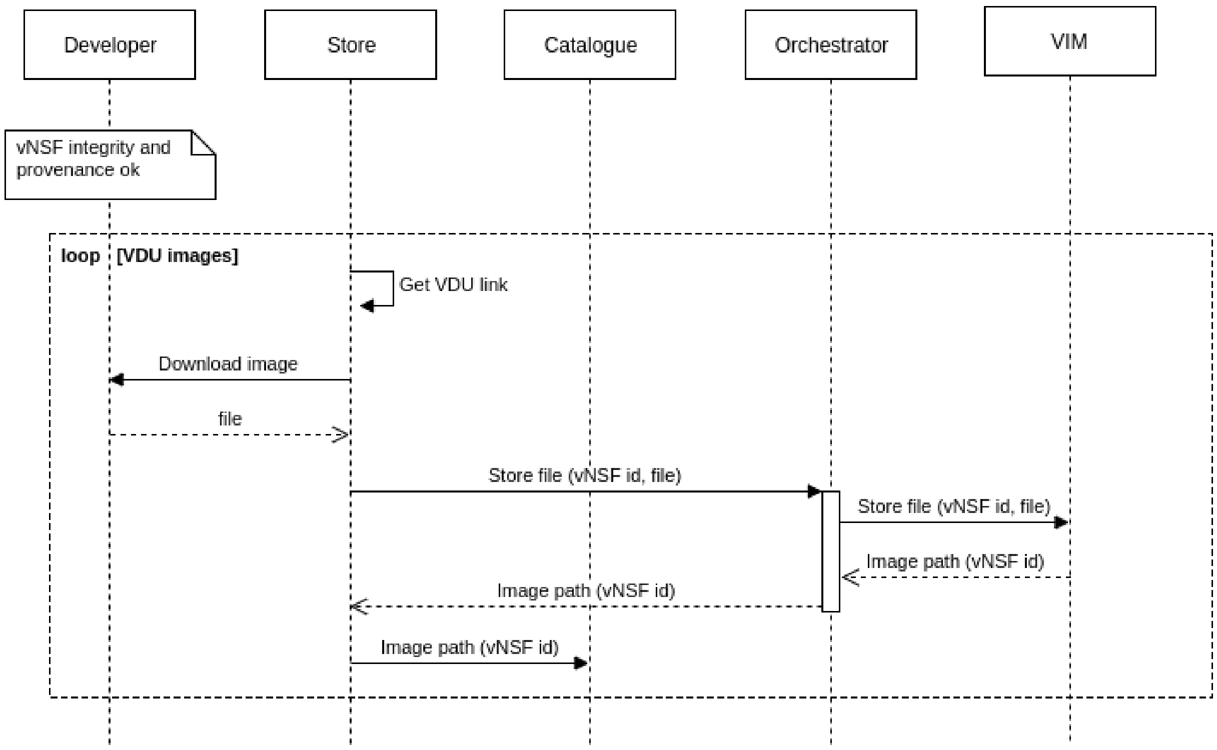


Figure 38: VDU image store.

NS/vNSF Decommissioning: When a NS or vNSF reaches the end of life it must be removed from the Store. This operation (Figure 39) is triggered by the Store which marks the NS or vNSF as decommissioned to prevent further instantiations. For a running NS or vNSF a graceful decommission is provided through the schedule of the operation to a later date.

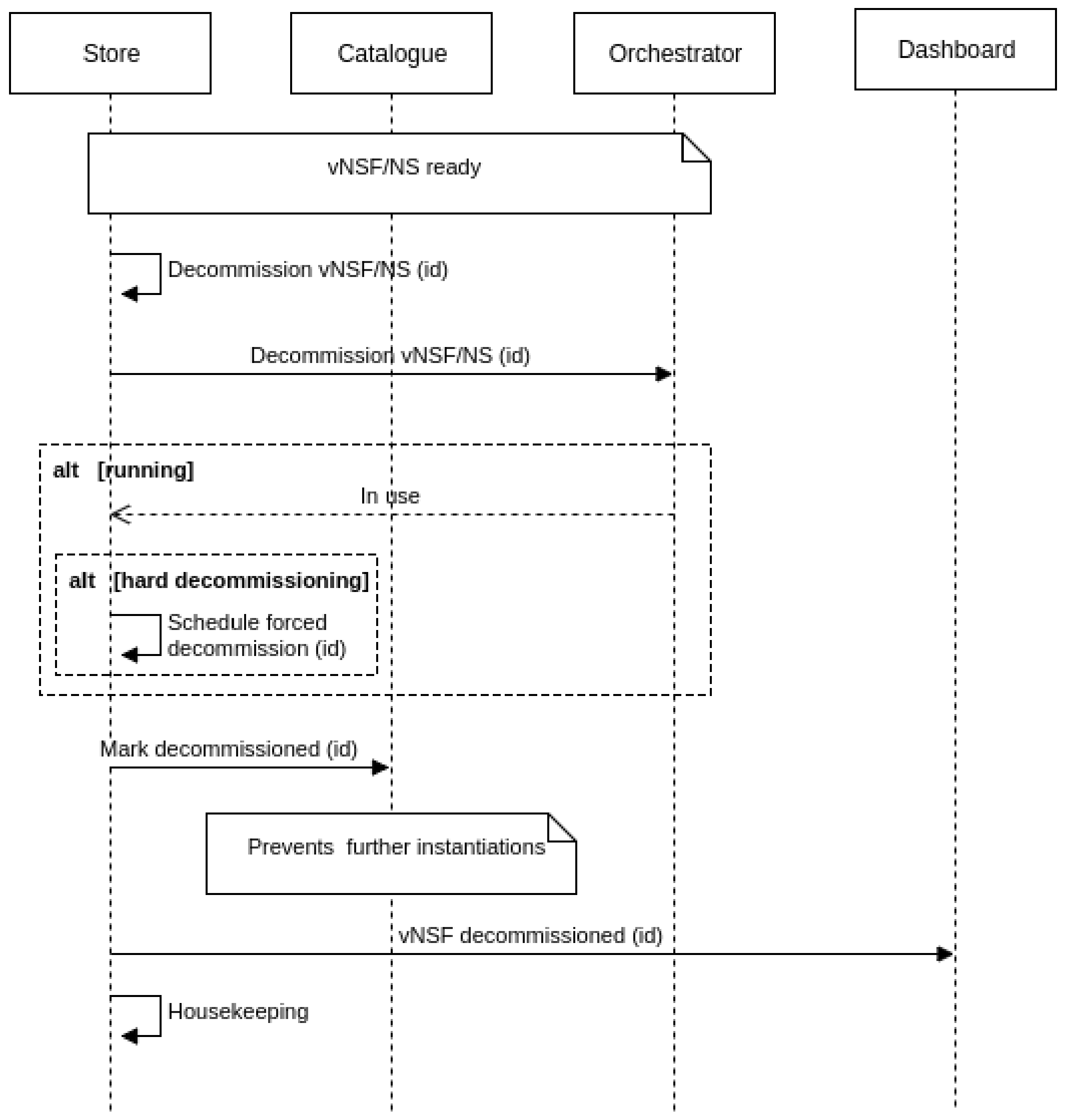

Store
Catalogue
Orchestrator
Dashboard
vNSF/NS ready
Decommission vNSF/NS (id)
Decommission vNSF/NS (id)
alt [running]
In use
alt [hard decommissioning]
Schedule forced decommission (id)
Mark decommissioned (id)
Prevents further instantiations
vNSF decommissioned (id)
Housekeeping

Figure 39: vNSF decommissioning.

vNSF Orchestrator

Interaction with the Store: The interaction between the Orchestrator and Store is effective during the deployment or instantiation. The vNSFO requests the NSD or vNSFD from the Store, as a first step to gather all resources for the NS instantiation, as depicted in Figure 33.

Interaction with the Network infrastructure: The vNSFO talks with the NFVI on every operation defined for the vNSF and NS Managers. It accounts for two type of operations: creating, updating or removing virtual links and fetching metrics from the infrastructure. The different interactions can be observed from Figure 33 to Figure 37.

Interaction with the Trust Monitor: The vNSFO will interact with the Trust Monitor at two points: first, when adding a physical node to the NFVI, so as to attest its software integrity before allowing it the access to the NFVI; and second, during the periodic attestation of the infrastructure. The process for the initial attestation is initiated by the vNSFO and is defined below, whereas the periodic attestation is depicted within the Trust Monitor section (Figure 34). This part of the process is depicted in Figure 30, and it is described as follows:

1. The vNSFO queries the Trust Monitor to attest a newcomer and provides information about the target
2. The Trust Monitor registers the node internally if not already there
3. For each node in the NFVI, the Trust Monitor establishes a Remote Attestation process
4. Each node of the NFVI sends back its integrity report to the Trust Monitor
5. The Trust Monitor assesses each integrity report by leveraging the list of known measurements in the whitelist, as well as expected dynamic configuration such as SDN forwarding rules
6. The Trust Monitor replies to the vNSFO with the attestation result (failure or success)

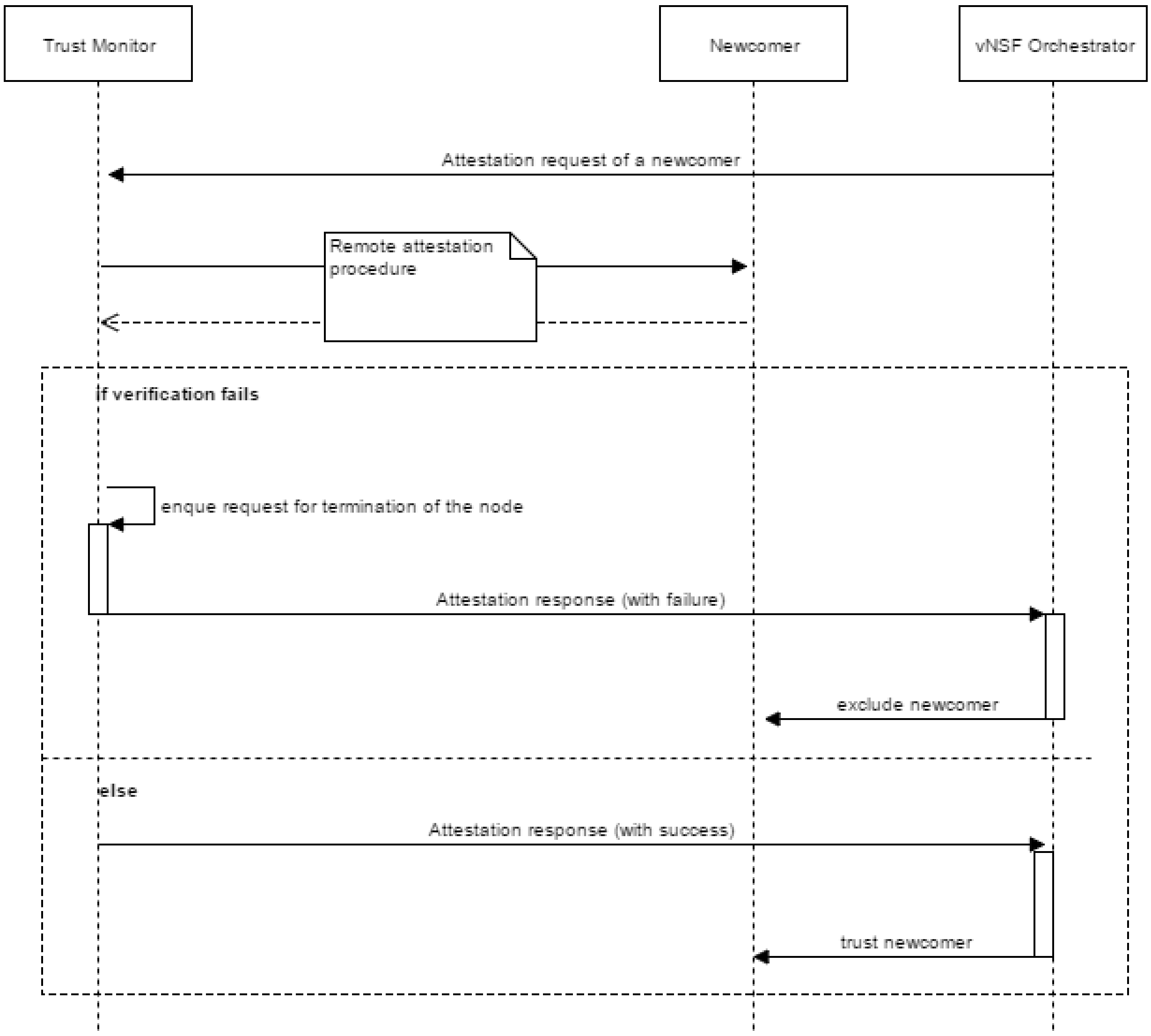

Trust Monitor
Newcomer
vNSF Orchestrator
Attestation request of a newcomer
Remote attestation procedure
If verification fails
enque request for termination of the node
Attestation response (with failure)
exclude newcomer
else
Attestation response (with success)
trust newcomer

Figure 40: Interaction between Trust Monitor and vNSFO in the initial attestation of a newcomer.

Interaction with DARE: The vNSFO provides the DARE with information on the network topology, the list of vNSFs per tenant and the running NSs and vNSFs. Such information is used by the subcomponents within DARE to analyse the most appropriate deployment to mitigate an active threat (Figure 41).

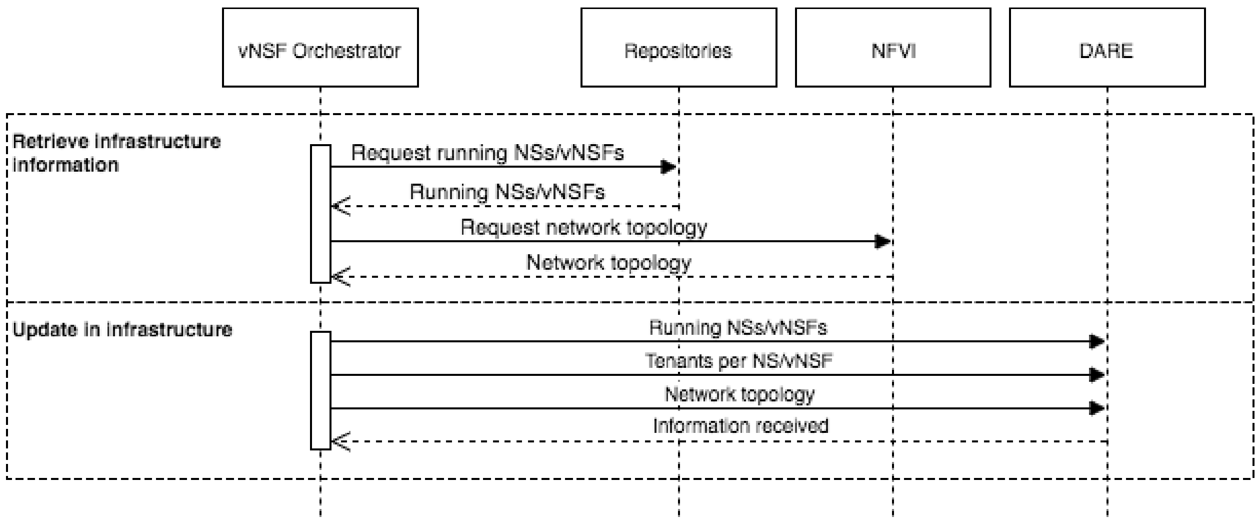


Figure 41: Interaction between the vNSFO and the DARE.

Interaction with Security Dashboard: The end-user will access the Security Dashboard to obtain relevant information about the infrastructure and possible suggestions to exert to mitigate a given threat. The interaction between the Security Dashboard and the vNSFO occurs at this point; where the suggestions are provided to the vNSF Orchestrator as a set of NSs to deploy, as well as the policies to provide to the specific constituent vNSFs at the deployed NSs (Figure 42).

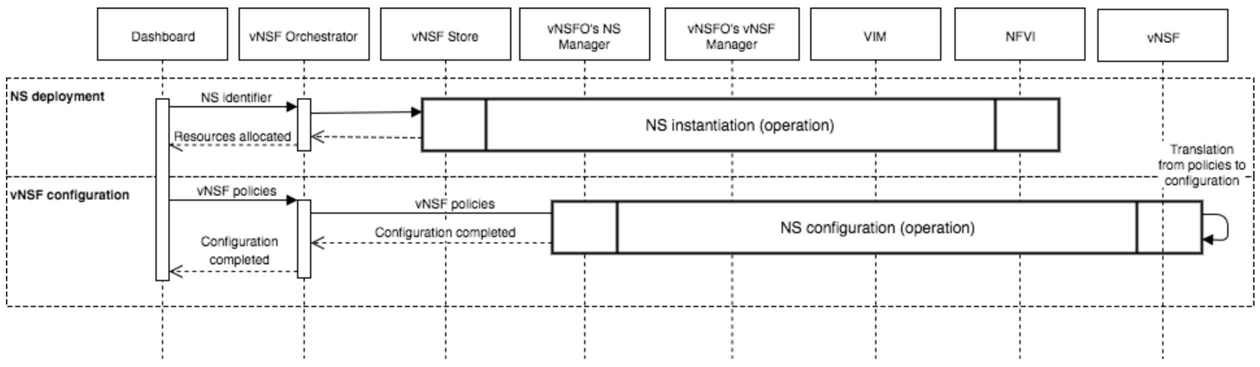


Figure 42: Interaction between the vNSFO and the Security Dashboard.

Trust Monitor

Interaction with Store: The Trust Monitor interacts with the Store to retrieve attestation-specific information needed to verify the integrity of the vNSFs running in the NFVI.

The process, pictured in Figure 43, is described as follows:

1. The Trust Monitor sends a request to the vNSF Store containing a specific vNSF identifier
2. The vNSF Store sends back a response with the requested vNSF's security manifest

3. The Trust Monitor extracts the measurements (digests) of the software executed by the vNSF
4. The Trust Monitor checks if the digests are included in the Whitelist Database
5. If no matching digest is present, the TM updates the whitelist with the new measurements and links them to the correct vNSF identifier

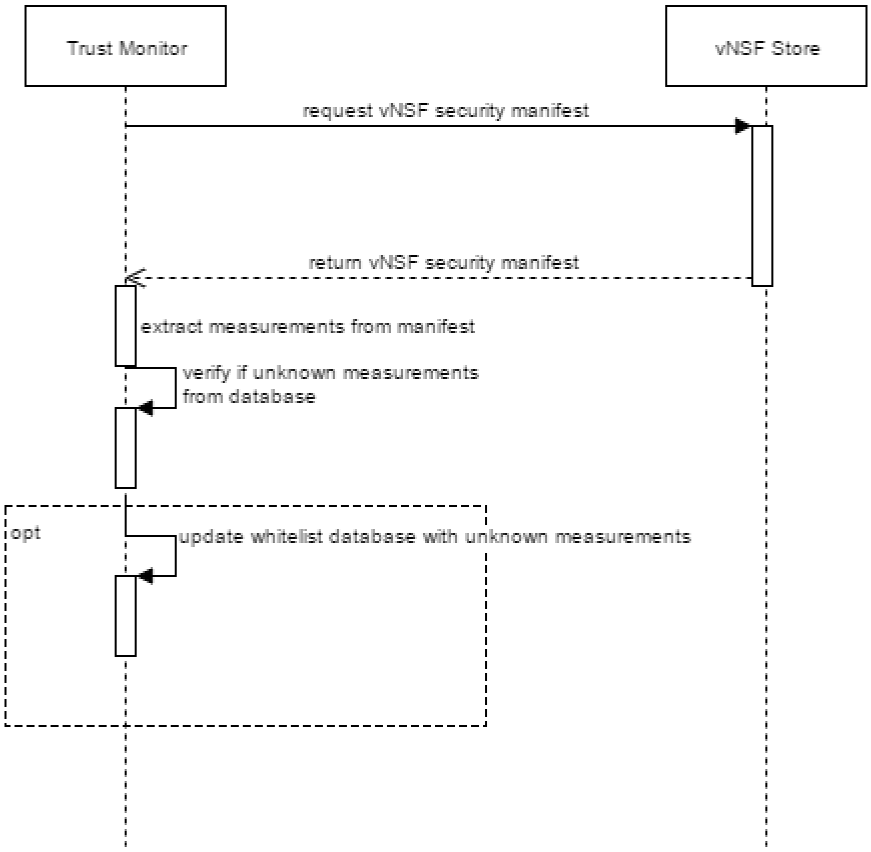

Trust Monitor
vNSF Store
request vNSF security manifest
return vNSF security manifest
extract measurements from manifest
verify if unknown measurements from database
opt
update whitelist database with unknown measurements

Figure 43: Interaction between Trust Monitor and vNSF Store.

Interaction with vNSF Orchestrator: The Trust Monitor interacts with the vNSFO when performing attestation of the NFVI; either on the initial attestation of a newcomer of the NFVI or during the periodic attestation task. The former process is described in the vNSF Orchestrator section (Figure 40), whereas the process for the periodic attestation is described below (Figure 44).

1. The Trust Monitor retrieves the NFVI state from the vNSFO
2. The Trust Monitor extracts the list of nodes to be attested from the NFVI
3. For each node in the NFVI, the Trust Monitor initiates a Remote Attestation procedure
4. Each node of the NFVI sends back its integrity report to the Trust Monitor
5. The Trust Monitor assesses each integrity report by leveraging the list of known measurements in the whitelist, as well as the expected dynamic configuration such as SDN forwarding rules
6. If any of the verifications fails:
    a. The Trust Monitor sends a notification about the failure to the vNSFO
    b. The vNSFO excludes the node from the NFVI
7. In the other case, the process successfully terminates

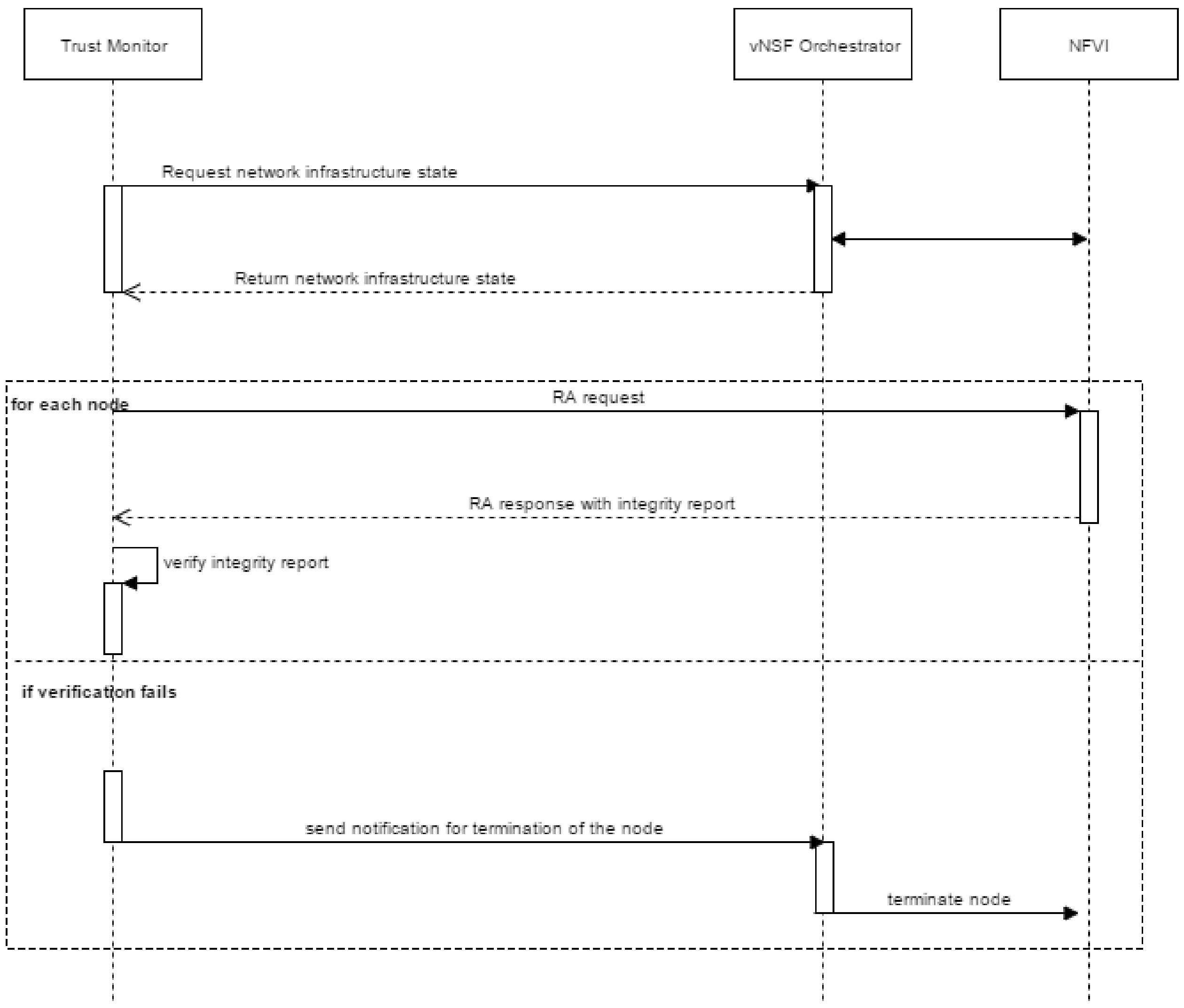

Trust Monitor
vNSF Orchestrator
NFVI
Request network infrastructure state
Return network infrastructure state
for each node
RA request
RA response with integrity report
verify integrity report
if verification fails
send notification for termination of the node
terminate node

Figure 44: Interaction between Trust Monitor and vNSF Orchestrator in the periodic attestation task.

Interaction with DARE: The Trust Monitor sends by sending security event information to the DARE (i.e., a node is found to be compromised during initial or periodic attestation tasks); this can then be processed by the Big Data engine for logging and further sense extraction thanks to its security modules. The workflow is depicted in Figure 45 and goes as follows:

1. The TM detects a security event that should be logged in the DARE, such as an attestation failure of a NFVI node or vNSF (either during initial or periodic attestation)
2. The TM sends the alarm to the DARE with the detailed information about the failure.

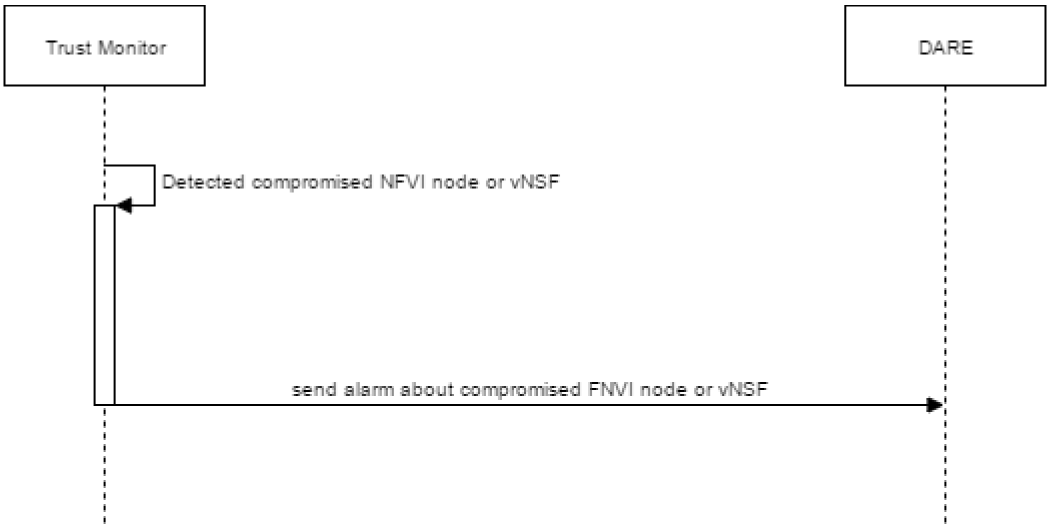

Trust Monitor
DARE
Detected compromised NFVI node or vNSF
send alarm about compromised FNVI node or vNSF

Figure 45: Interaction between Trust Monitor and DARE.

ANNEX C. TRUSTED COMPUTING TECHNOLOGIES

Trusted Computing aims at providing specific technologies and mechanisms to establish a hardware-based assessment of the integrity of a computing system. The Trusted Computing Group (TCG) [26] is the major company-backed TC consortium, which mainly focuses on the development of solutions for enabling TC in computing platform from mobile and embedded devices to data-centre class servers.

One of the fundamental principle of TC is the Chain of Trust (CoT), a transitive mechanism that ensures the trustworthiness of a computing system via a step-by-step extension process. The process requires the definition of a minimal combination of hardware and software - called Core Root of Trust for Measurement (CRTM) - that initiate the CoT measuring - and storing the measurement - the next software to be executed; it is implicitly trusted by a remote verifier. Each element of the CoT is responsible for measuring and storing the integrity of the next element, so that the whole chain can be verified by a third party. The starting point of the verification process is the CRTM, whose establishment requires a dedicated hardware security chip, called Trusted Platform Module (TPM).

The TPM is a device, standardised by the TCG, acting as a secure cryptoprocessor capable of storing keys, secrets, identities and measurements of the platform integrity. The standard has undergone different revisions, reaching the 2.0 version at the time of writing. Integrity measurements are protected by the TPM’s Platform Configuration Registers (PCR). PCRs can only be updated by the TPM itself, using an internal secure hash function, via the “extend” operation: at each step, the current value of a PCR is concatenated with the new measurement and the digest of the resulting message is stored in the PCR. This mechanism ensures that unless the platform is rebooted, no PCR-stored measurement can be erased - thus software-based attacks cannot hide execution of untrusted binaries.

The TCG also defines a specific workflow to attest the trustworthiness of TPM-equipped and measured boot enabled entities by a remote third party, called Remote Attestation. The PCRs’ value can be accessed by a remote entity by challenging the TPM with a nonce; using a hardware-protected key (i.e. only the TPM can use the private key for signing), the TPM protects the integrity of the PCRs’ with a signature which include the challenge nonce for freshness. Using the prior knowledge of all the platform’s TPM public key used for attestation, the remote entity can verify the genuineness of the signature - which also validates the hardware identity, as well as the content of the logged software events.

The TPM specification does not specify the measurement strategy to be adopted by the computing system for logged software events. The Integrity Measurement Architecture (IMA) [27] in Linux is a specific implementation that maintains a log of measured software events (e.g. the execution of a binary, using a configuration file) at runtime and, if enabled with a TPM, an aggregate integrity value is stored in one of the static PCRs. Although the log file might be manipulated by an attacker, the hardware register can’t be directly altered, meaning that a verifier could detect any unexpected tampering to the log file.

ANNEX D. APPLICATION PROGRAMMING INTERFACES (APIS)

This Annex presents a first definition of the methods (and arguments) to be supported by the APIs exposed by each component.

Orchestrator

This section includes the low-level specifications of the operations offered by each API exposed by the vNSFO.

Dashboard API: The vNSFO will provide an interface so that the Security Dashboard can retrieve the necessary information to provide the visualisation to the end-user.

Operation	Arguments	Description
get_network_topology	-	Provides the topology of the network as provided by the VIM
get_deployed_vnsfs	-	Provides the running vNSFs
get_deployed_vnsfs	tenant_id	Provides the running vNSFs, filtered by tenant

Trust Monitor API: The vNSFO will provide an interface so that the Trust Monitor can obtain the information to perform the periodic attestation task.

Operation	Arguments	Description
get_physical_nodes	-	Provides the list of active physical nodes in the NFVI
get_deployed_vnsfs	-	Provides the running vNSFs
get_network_topology	-	Provides the topology of the network as provided by the VIM
get_network_flowtable	-	Provides the contents of the flow tables of the SDN controller

DARE API: The vNSFO will provide an interface for the DARE to obtain a global view on the NFVI and thus be able to perform the analytics and provide the recommendations.

Operation	Arguments	Description
get_network_topology	-	Provides the topology of the network as provided by the VIM
get_deployed_vnsfs	tenant_id	Provides the running vNSFs, filtered by tenant
get_deployed_vnsfs	-	Provides the running vNSFs

get_deployed_nss	-	Provides the running NSs

Trust Monitor

This section includes the low-level specifications of the operations offered by each API exposed by the Trust Monitor.

Management API: The Trust Monitor will provide a Management API with operations that would allow other components to check the status of the infrastructure's attestation.

Operation	Arguments	Description
get_status_info		Retrieves status information about the Trust Monitor
get_vnsf_attestation_info	node_id	Retrieves attestation-specific information for a single vNSF
get_nfvi_attestation_info		Retrieves attestation-specific information for the whole NFVI
get_nfvi_pop_attestation_info	node_id	Retrieves attestation-specific information for a specific NFVI PoP

Newcomer Attestation API: The Trust Monitor will provide an interface for receiving attestation requests for a newcomer in the NFVI. Note that the interface should be specific for the newcomer's attestation, as the Trust Monitor will later on perform periodic attestation tasks over the different nodes that have been pre-registered to it. The API may be used for both physical nodes, during the initial authentication phase, or for vNSFs, during their instantiation phase.

Operation	Arguments	Description
register_node	node_id, address, distribution, ...	Registers the node to the Verifier, given a unique identifier (which will be used for further attestation procedure), the address of the node, the distribution of the OS running in it.
attest_node	node_id, analysis_type	Remote Attestation request to the node, identified by a unique ID. The client to be attested will provide the integrity measurements according to the type of requested analysis (e.g. load-time analysis with a certain trust level for the measurements)

Store

The Store will provide an interface to obtain the information it persists as well as accessing features it provides.

Operation	Arguments	Description
onboard_vnsf	security_manifest, vnsf_descriptor	Onboards a vNSF
onboard_ns	security_manifest, ns_descriptor	Onboards a NS
get_vnsf_onboarding_status	id	Provides the status for the vNSF onboarding operation
get_ns_onboarding_status	id	Provides the status for the NS onboarding operation
list_vnsfs	-	Provides a list of all the onboarded vNSFs along with a brief description for each one
list_nss	-	Provides a list of all the onboarded NSs along with a brief description for each one
get_vnsf_info	id	Provides all the information on the onboarded vNSF
get_ns_info	id	Provides all the information on the onboarded NS
decommission_vnsf	id	Retire a vNSF
decommission_ns	id	Retire a NS
get_vnsf_security_info	id	Provides all the security information concerning a vNSF
get_ns_security_info	id	Provides all the security information concerning a NS

ANNEX E. TECHNOLOGY SELECTION

Orchestrator

The analysis of different vNSFOs has been carried out to choose which one to use in SHIELD. To do this we selected a subset of some well-known open-source NFV MANO (TeNOR, OSM, SONATA, OpenBaton) and considered the adequateness depending on the mapping to the SHIELD's Platform Functional Requirements, the support of some relevant key features within the project and the status of its community and development.

Platform Functional mapping

PF Requirement	TeNOR	OSM	SONATA	OpenBaton
PF01 - vNSF and Network Service (NS) deployment	Y	Y	Y- (No external cloud deployment)	Y
PF02 - vNSF lifecycle management (on boarding, instantiation, chaining, configuration, monitoring and termination)	Y	Y- (Monitoring based on VIM implementation integrated in R3)	Y	Y
PF03 - vNSF status management (DEPLOY, START, STOP, MODIFY, DELETE)	Y	Y	Y- (Ongoing for: adding restart, stop, pause)	Y- (Some may be missing)
PF04 - Security data monitoring and analytics	Y- (Delegated to NSM and vNSFM)	Y- (Delegated to the EM)	Y- (Custom metrics allowed, infra metrics)	Y-
PF05 - Analytics visualisation	N/A	N/A	N/A (son-gui shows monitoring metrics)	N/A
PF06 - Ability to offer different management roles to several users (multi-user with possibility of configuring different roles)	N	N	Y (Static dev/customer roles; new roles will be customised and dynamic)	N
PF07 - Service elasticity [optional]	Y	Y (Experimental NS scaling. manual GUI, support for adding/removing full VNFs to/from a running NS)	Y- (Will allow scale-out)	Y

PF08 - Platform expandability	Y	Y	Y	Y
PF09 - Access control	Y (Tokens)	Y (Certificates)	Y (User/sw, sw-sw using tokens)	Y
PF10 - vNSF validation	N/A	N/A	N (Signed packages in store, control mangling)	N/A
PF11 - vNSF attestation	N	N	N	N
PF12 - Log sharing	N/A	N/A	N/A	N/A
PF13 - Mitigation	Y	Y	N	Y
PF14 - Multi-tenancy	Y (1 tenant: 1 running NS)	Y-	N	N (1 tenant for all)
PF15 - Service store	N/A	N/A	Y	N/A
PF16 - History reports	N/A	N/A	N (alerts aggregated)	N/A
PF17 - Interoperability	N/A	Y (Multiple VIMs)	N	N/A
PF17 - Interoperability	Y	Y	Y	Y
PF19 - Network infrastructure attestation	N/A	N/A	N	N/A
PF20 - Billing framework	N (Delegated to BSS)	N (Delegated to BSS)	N/A (License concept in NS related to billing)	N

Feature-focused analysis

Feature	TeNOR	OSM	SONATA	OpenBaton
Type of virtualisation	VMs	VMS (Containers may be possible)	VMs	VMs
VIM supported	OpenStack	OpenVIM (R1/R2), OpenStack (R2), VMWare (R2)	OpenStack	OpenStack
SDN controller supported	ODL (through netfloc)	ODL (R1/R2), Floodlight (R1/R2), ONOS (R2)	ODL	ODL, ONOS (ongoing)
Service Function Chaining	Y (Using netfloc plug-in and ODL)	Y (Direct, no plug-in)	Y (ODL SFC)	Y

Lifecycle management	Y (Start, stop)	Y (Available for VNF & NS)	Y	Y
Event management	Y (Custom)	Y (Provide messages about the deploy of vNSF & NS)	Y- (ongoing)	Y (Generic and specific)
Elasticity	Y (Scale-in, scale-out)	Y (Scale-in, scale-out; experimental support to modify running NSs)	Y- (Will allow scale-out)	Y (Auto-scaling)
Monitoring	Y (Per NS, per vNSF instance and inner service)	Y (Based on VIM, delegated to EM)	Y (Prometheus, log aggregation per component)	Y (Zabbix for NFVI and VNFs)
Dynamic vNSF placement	Y- (Algorithms in place, not tested)	N	N (Ongoing design for auto-location and distributed NSs)	Y

Maintenance-focused analysis

Key	TeNOR	OSM	SONATA	OpenBaton
LoC	441217	340861	6596	118322
Development language	Ruby	Python	Ruby, Python	Java
Community	i2CAT	ETSI and 60 orgs (8 net operators)	ATOS, i2CAT, etc	Fraunhofer/FOKUS, TUB
Projects in use	EU and national R&D ongoing projects	EU R&D projects, Telefónica VNF cert Lab, RIFT.ware	EU R&D projects	EU R&D project, 5GBerlin testbed

ANNEX F: SHIELD PACKAGING

vNSF Packaging

Elements

To foster VNF reuse and remove SHIELD applicability barriers the SHIELD VNF package format extends existing VNF formats by introducing:

- a security manifest to ensure VNF tamper-proofing
- a digitally-signed security manifest to prove provenance and integrity
- support for including Orchestrator-specific VNF package format
- a .tar.gz package format to enclose everything

A SHIELD vNSF package (.tar.gz file) comprises:

Table 22 vNSF package

Element	Format	Purpose
manifest.yaml	YAML	Security manifest which defines the tamper-proof metadata to ensure the vNSF in operation wasn't tampered with since when it was onboarded
<vnf_package_file>	Orchestrator specific	The VNF package to onboard into the vNSF Orchestrator

Structure

The structure of a SHIELD vNSF package is as follows:

```
.
├── manifest.yaml           # SHIELD security manifest
└── <vnf_package_file>      # Orchestrator-specific VNF package
```

This packaging is Orchestrator agnostic and allows for onboarding an existing VNF into SHIELD simply by providing a security manifest tailored to the VNF in question. Once this is done it is just a matter of producing a .tar.gz file with the contents mentioned and submit it to the Store.

Datamodel

The security manifest (`manifest.yaml`) datamodel is described in Table 23.

Table 23 vNSF manifest datamodel

Field	Purpose
`manifest:vnsf`	Defines a SHIELD vNSF package
`type`	The type of VNF the manifest describes. Allowed values: `OSM`
`package`	VNF file name within the SHIELD package. This file name, contents and format is Orchestrator specific. This manifest only identifies the file which holds the VNF package
`hash`	The message digest for the VNF package mentioned in the `package` field.
`descriptor`	VNF Descriptor file within the VNF-specific package. Tipically a path to the actual file itself
`properties`	vNSF characterization and purpose-related details
`security_info`	The metadata used for attestation purposes to ensure the VNF wasn't tampred with

Schema

```
'schema': {
  'manifest:vnsf': {
    'type': 'dict',
    'required': True,
    'schema': {
      'type': {
        'type': 'string',
        'empty': False,
        'allowed': ["OSM"],
        'required': True
      },
      'package': {
        'type': 'string',
        'empty': False,
        'required': True
      },
      'hash': {
        'type': 'string',
        'empty': False,
        'required': True
      },
      'descriptor': {
        'type': 'string',
        'empty': False,
        'required': True
      },

      #vNSF description.
      'properties': {
        'type': 'dict',
        'required': True,
        'schema': {
          'vendor': {
            'type': 'string',
            'empty': False,
            'required': True
          },
          'capabilities': {
            'type': 'list',
            'empty': False,
            'required': True
          },
        }
      },

      'security_info': {
        'type': 'dict',
        'required': True,
        'schema': {
          'vdu': {
            'type': 'list',
            'required': True,
            'schema': {
              'type': 'dict',
              'schema': {
                'id': {
                  'type': 'string',
                  'empty': False,
                  'required': True
                },
                'hash': {
                  'type': 'string',
                  'empty': False,
                  'required': True
                },
                'attestation': {
                  'type': 'dict',
                  'required': True,
                  'schema': {
                    'somekey': {
                      'type': 'string',
                      'empty': False,
                      'required': True
                    }
                  }
                }
              }
            }
          }
        }
      }
    }
  }
}
```

Example: OSM VNF packaging

The SHIELD vNSF package extends the OSM VNF package definition by adding the security details. The SHIELD vNSF package is thus a wrapper that contains the elements in Table 24.

Table 24 OSM-based vNSF package example

Element	Contents [(M)andatory | (O)ptional]	Source
manifest.yaml	(M) package contents definition along with the security information	SHIELD
<*vnf_name*>_vnfd.yaml	(M) vNSF descriptor information. Follows the OSM Information Model (page 47)	OSM
charms	(O) juju charm configuration for the VNF	OSM
checksums.txt	(M) image file(s) hash(es)	OSM
cloud_init	(O) instantiation configurations	OSM
icons	(O) used on the OSM Composer	OSM
images	(O) VDU image files for the vNSF	OSM
README	(O) vNSF related information	OSM
scripts	(O) base configuration scripts once the vNSF is up and running	OSM

The security manifest for an OSM VNF would look like this:

```
manifest:vnsf:
  type: OSM
  descriptor: cirros_vnf/cirros_vnfd.yaml
  package: cirros_vnf.tar.gz
  hash: bVa7rrrYocuye2TMbUEk+8NXUPRJ6TYhfRWip66kfO0=
  properties:
    vendor: some vendor name
    capabilities: ['Virtual Cirr OS']
  security_info:
    vdu:
      - id: cirros_vnfd-VM
        hash: KFQDLBG64WUZZKd/RSetWD/ymcyn4hcOn9oFpMd6rLE=
        attestation:
            somekey: <TBD - provided by TM>}
            somekey: <TBD - provided by TM>}
```

NS Packaging

Elements

To foster Network Service reuse and remove SHIELD applicability barriers the SHIELD Network Service package format extends existing formats by introducing:

- a digitally-signed security manifest to prove provenance and integrity
- support for including Orchestrator-specific Network Service package format
- a .tar.gz package format to enclose everything

A SHIELD NS package (.tar.gz file) comprises:

Table 25 Network Service package

Element	Format	Purpose
manifest.yaml	YAML	Security manifest which defines the tamper-proof metadata to ensure the Network Service in operation wasn't tampered with since when it was onboarded
<ns_package_file>	Orchestrator specific	The Network Service package to onboard into the Network Service Orchestrator

Structure: The structure of a SHIELD Network Service package is as follows:

```
.
├── manifest.yaml           # SHIELD security manifest
└── <ns_package_file>       # Orchestrator-specific Network Service package
```

This packaging is Orchestrator agnostic and allows for onboarding an existing Network Services into SHIELD simply by providing a security manifest tailored to the Network Service in question. Once this is done it is just a matter of producing a.tar.gz file with the contents mentioned and submit it to the Store.

Datamodel: The security manifest (`manifest.yaml`) datamodel is described in Table 26.

Table 26 Network Service manifest datamodel

Field	Purpose
`manifest:ns`	Defines a SHIELD Network Service package
`type`	The type of VNF the manifest describes. Allowed values: `OSM`

`package`	Network Service file name within the SHIELD package. This file name, contents and format is Orchestrator specific. This manifest only identifies the file which holds the Network Service package
`hash`	The message digest for the NS package mentioned in the `package` field.
`descriptor`	Network Service Descriptor file within the Network Service-specific package. Tipically a path to the actual file itself
`properties`	Network Service characterization and purpose-related details

Schema

```
'schema': {
  'manifest:ns': {
    'type': 'dict',
    'required': True,
    'schema': {
      'type': {
        'type': 'string',
        'empty': False,
        'allowed': ["OSM"],
        'required': True
      },
      'package': {
        'type': 'string',
        'empty': False,
        'required': True
      },
      'hash': {
        'type': 'string',
        'empty': False,
        'required': True
      },
      'descriptor': {
        'type': 'string',
        'empty': False,
        'required': True
      },

      #NS description.
      'properties': {
        'type': 'dict',
        'required': True,
        'schema': {
          'vendor': {
            'type': 'string',
            'empty': False,
            'required': True
          },
          'capabilities': {
            'type': 'list',
            'empty': False,
            'required': True
          },
        }
      },
    }
  }
}
```

Example: OSM NS packaging

The SHIELD Network Service package is a wrapper that contains the elements described in Table 27.

Table 27 OSM-based NS package example

Element	Contents [(M)andatory | (O)ptional]	Source
manifest.yaml	(M) package contents definition along with the security information	SHIELD
<*ns_name*>_nsd.yaml	(M) Network Service descriptor information. Follows the OSM Information Model (page 12)	OSM
checksums.txt	(M) image file(s) hash(es)	OSM
icons	(O) used on the OSM Composer	OSM
README	(O) vNSF related information	OSM

The security manifest for an OSM NS would look like this:

```
manifest:ns:
    type: OSM
    package: cirros_ns.tar.gz
    hash: qrv4hyEUbgcU7j(zkepfubmdotU69436bkbndRGUUo0=
    descriptor: cirros_ns/cirros_nsd.yaml
    properties:
        capabilities: ['Virtual Cirr OS']
```

ANNEX G. TECHNICAL UPDATES AND REVISIONS

Technical advances and updates since D2.1/D3.1

vNSFO: The subcomponents of the vNSFO have been further analysed against the requirements, and consequently adapted. This analysis resulted in some subcomponents being renamed to better state their functionality (the "Orchestrator Engine" has been renamed to "NS Manager", the "Infrastructure Repository" was renamed to "Repositories", "Northbound" and "Data engine" APIs were renamed to "Store", "Dashboard", "DARE" Connectors and APIs). Some were added to cover extra functionalities (the connection with the Trust Monitor) and others were removed to define the architecture more clearly and avoid duplicities (the "Catalogue", introduced as a separate subcomponent, is kept in the Store; and the "Monitoring" is a feature provided by the NS and vNSF Managers). The mapping of the functionality of the vNSFO with the requirements from SHIELD is revised according to the specifications proposed. "PF10 - vNSF validation" is now a responsibility of the Store. On the other hand, other Platform Functional (PF) requirements and Non-Functional (NF) requirements have been now mapped to vNSFO to cover related, though not direct, responsibilities. After further analysis, the interface between the Trust Monitor and the vNSFO should be extended to support enrolment of a newcomer node on the NVFI PoP in the TM.

DARE/Trust Monitor: The mapping between the component and the Platform Requirements (PF), as envisioned in D2.1, has been reconsidered with regards to the capabilities of the other components of the platform. More specifically, "PF04 - Security data monitoring and analytics" is addressed by the data acquisition and analysis capabilities provided by the DARE, as the Trust Monitor does not receive logs straight from the vNSFs to detect occurring security incidents. The "PF13 - Mitigation" requirement is addressed by the recommendation and remediation capabilities of the DARE; the corresponding requirement for the Trust Monitor is "PF19 - Network infrastructure attestation". The "PF18 - Service composition" requirement is addressed by the Security Dashboard and the DARE, as they are the components involved in the selection and deployment of vNSFs. Differently from D2.1, the "PF11 - vNSF attestation" requirement is fulfilled by the Trust Monitor, which is collecting attestation's data from the hosts running the vNSFs and check their integrity information against the known values retrieved from the Store.

Store: As a result of the specification activities, it was decided to keep a single NS/vNSF catalogue instance for use in the SHIELD platform, placing this catalogue as a subcomponent of the Store. This approach helps to reduce information replication throughout SHIELD's components and define more clearly the responsibilities of both Store and vNSFO components. The Store will be responsible for managing and providing the information of all the onboarded NSs and vNSFs. Additionally, after further analysis of the requirements in D2.1, two additional platform requirements were mapped to the Store component: "PF02 - vNSF lifecycle management" was included since the Store is responsible for managing partially the lifecycle of vNSFs, being responsible for the onboarding process; "PF11 - vNSF attestation" is now mapped to the Store, since the Store will validate the digital-signature of each artefact onboarded, thus assessing the validity of its provenance.

SHIELD vNSFs: The scope of the vDPI was altered since D2.2 to account for user case 3 and the input provided by cybersecurity agencies.

Document revision

The following table tracks the changes to this document, as compared to its preliminary (base) version, D3.1.

#	Page No	Revision Date	Revision action	Revision description	Tracking notes
1	58-86	20/2/2018	Added	Chapter 4, "Regulatory Compliance Specifications"	Added to ensure GDPR compliance for vNSFs, and to address comments during Y1 review
2	87-96	23/2/2018	Added	Chapter 5, "Validation and Certification"	Added to discuss performance and testing of vNSFs, certification options, marketplace development and to address comments during Y1 review
3	2-3	2/3/2018	Updated	Executive Summary	Executive summary reflects the new requirements in D2.2 and the scope of the deliverable w.r.t. D3.1
4	8-9	2/3/2018	Added	Subsections 1.2 Scope of this deliverable, 1.3 Organisation of this document	Added to provide the scope of the document, links to other deliverables and the organisation of its contents
5	10-11	2/3/2018	Added	Subsection 2.1 Guiding Principles	Addresses standardisation, legal compliance and exploitation and basic design principles, text has been separated in a new subsection.
6	-	2/3/2018	Removed	2.2.2, 2.3.2, 2.4.2 Updates since D2.1	This text was removed as it was deprecated. Technical Updates w.r.t. D2.2/D3.1/D4.1 are included in the present Annex.
7	24	23/3/2018	Updated	2.2.2.4 Functionality Mapping	Updated to include all requirements in D2.2 and Map the vNSF ecosystem functionalities to the demos/use cases.
8	36-57	23/3/2018	Updated	3.1.x.1 Implementation details	Finalised implementation details for Y1 vNSFs (vDPI, vIDPS)
9	36-57	23/3/2018	Added	3.1.x.1 Implementation details	added implementation details for Y2 vNSFs (proxyTLS, HTTP/S analyser, L3 filter, forward L7 filter)
10	36-57	23/3/2018	Updated	3.1.x.2 Requirements Mapping	Updated to include the new requirements in D2.2, and resource requirements.
11	96-97	23/3/2018	Updated	Section 6 Conclusions	Updated section number and includes future work for WP3 and conclusions/lessons learned after Y1
12	139-140	26/3/2018	Added	Annex F	Tracks changes in technical content and in the text since D3.1.